VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2018 Bc. Tomáš Kloda

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

APLIKACE PRO HROMADNOU SPRÁVU SÍŤOVÝCH PRVKŮ MIKROTIK
APPLICATION FOR BATCH MANAGEMENT OF MIKROTIK NETWORK DEVICES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Tomáš Kloda**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Ondřej Krajsa, Ph.D.**
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Tomáš Kloda ***ID:*** 154767
Ročník: 2 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Aplikace pro hromadnou správu síťových prvků Mikrotik

POKYNY PRO VYPRACOVÁNÍ:

Vytvořte interaktivní aplikaci pro hromadnou správu sítě založené na aktivních prvcích Mikrotik. Aplikace bude využívat Mikrotik API-SSL, uživatelské rozhraní bude realizováno webovou aplikací.

DOPORUČENÁ LITERATURA:

[1] BURGESS, Dennis. Learn RouterOS. Lexington]: Dennis Burgess, 2009, 391 s. : il. ISBN 978-0-557-09271-0.

[2] LOCKHART, Josh. Modern PHP: New Features and Good Practices. Sebastopol: O'Reilly Media, 2015. ISBN 9781491905012.

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 21.5.2018

Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Tato práce je zaměřena na popis a vytvoření webové aplikace určenou pro hromadnou správu síťových prvků firmy MikroTik. V teoretické části se práce soustředí na popis operačního systému RouterOS. Dále se práce zabývá popisem komunikace přes službu MikroTik API. Další část zahrnuje stručný popis nástrojů, s jejíž pomocí je webová aplikace tvořena. V praktické části je pak popsán vývoj a realizace webové aplikace. Webová aplikace je určena pro operační systém Linux.

KLÍČOVÁ SLOVA

API, API-SSL, hromadná správa, MikroTik, RouterOS, webová aplikace

ABSTRACT

The goal of the thesis is to describe and develop web application for large-scale management of MikroTik network devices. The theoretical part describes RouterOS operating system. This work also describes communication through MikroTik API service. Next section includes brief description of tools, which were used for the web application development. The practical part describes development and realization of the web application. The web application is intended for Linux based operating systems.

KEYWORDS

API, API-SSL, large-scale management, MikroTik, RouterOS, web application

KLODA, Tomáš. *Aplikace pro hromadnou správu síťových prvků MikroTik*. Brno, 2017, 56 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Aplikace pro hromadnou správu síťových prvků MikroTik“ jsem vypracoval(a) samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Ondřeji Krajsovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Telekomunikační infrastruktury podniků se neustále vyvíjí. Většina firem je nucena řešit problém, jak nejvíce zefektivnit a zjednodušit správu svých telekomunikačních sítí. Mnoho malých i středně velkých firem rovněž používá ke stavbě své telekomunikační infrastruktury síťové prvky MikroTik.

Tato práce se zabývá vývojem interaktivní webové aplikace, která umožní zjednodušit a urychlit správu telekomunikační infrastruktury postavené na síťových prvcích firmy MikroTik. V úvodu teoretické části je práce zaměřena na popis operačního systému RouterOS, kde jsou stručně vysvětleny veškeré možnosti konfigurace. Dále se teoretická část soustředí na detailní popis aplikačního rozhraní firmy MikroTik. Je zde vysvětlen princip komunikace a význam jednotlivých slov, které jsou pro komunikaci s RouterOS používány. V závěru teoretické části jsou stručně popsány jednotlivé nástroje, které jsou použity pro vývoj webové aplikace.

Praktická část práce již řeší samotný vývoj webové aplikace a je rozdělena do dvou částí. První část se zabývá síťovými prvky, které se nacházejí v továrním nastavení. Je zde popsáno řešení pro nalezení všech síťových prvků v síti a zajištění základní konektivity na síťové vrstvě. Rovněž je zde popsána problematika certifikátů a možnost třídění nalezených síťových prvků do skupin.

Druhá část se pak zabývá síťovými prvky, které jsou již schopné komunikovat na síťové vrstvě. Je zde popsáno řešení hromadné konfigurace jednotlivých síťových prvků v lokální síti. Taktéž je zde popsána funkce hromadných záloh konfiguračních souborů a možnosti hromadného nahrávání výchozích konfiguračních souborů pro jednotlivé skupiny síťových prvků. Rovněž je zde popsána implementace monitorovacího systému a vykreslování diagramu síťové topologie. Na konec je provedeno porovnání komunikace pomoci běžné a šifrované varianty aplikačního rozhraní.

V závěru praktické části se nachází popis jednotlivých rozhraní webové aplikace. Je zde vysvětleno, jak se ve webové aplikaci orientovat a jakým způsobem ji ovládat.

1 MIKROTIK

MikroTik je lotyšská společnost zabývající se vývojem softwarových a hardwarových řešení pro internetové připojení. Mezi hlavní produkty firmy patří především směrovače RouterBoard a bezdrátové systémy. Firma byla založena v roce 1996 [1].

1.1 RouterOS

RouterOS je proprietární operační systém firmy MikroTik, který umožňuje běžným hardwarovým zařízením (např. jakémukoliv počítači) vykonávat funkce plnohodnotného směrovače. Mezi tyto funkce lze zařadit směrování paketů, řízení šířky pásma, řízení toku dat, virtuální privátní sítě, přístupové body a mnoho dalších [2].

RouterOS je založen na operačním systému Linux a je kompatibilní se systémy běžícími na platformě x86 a se zařízeními MikroTik RouterBoard, na kterých je RouterOS standardně předinstalován. RouterOS je velmi vhodný pro nasazení do domácností či do středně velkých podniků. Flexibilita a jeho široké možnosti nasazení umožňují snížit celkové náklady na hardware a jeho údržbu [2].

1.1.1 Možnosti konfigurace

RouterOS nabízí poměrně rozsáhlé možnosti přístupu a konfigurace. V případě přímé konfigurace se lze k síťovému prvku připojit pomoci sériového kabelu. Vzdálenou konfiguraci RouterOS lze provést prostřednictvím nešifrovaného protokolu Telnet či šifrovaného protokolu SSH (Secure Shell) [2].

Jednou z nejsilnějších výhod RouterOS je schopnost vzdálené konfigurace na druhé (linkové) vrstvě ISO/OSI modelu (tzv. MAC-Telnet). Ke vzdálenému připojení na zařízení je potřeba znát pouze jeho MAC (Media Access Control) adresu. Tato metoda je velmi výhodná pro základní konfiguraci síťového prvku, který se nachází v továrním nastavení a tudíž nemá nastavenou IP (Internet Protocol) adresu na žádném rozhraní. Metoda MAC-Telnet bude využita v rámci implementace webové aplikace [9].

Další metodou vzdálené konfigurace je proprietární aplikace WinBox, která s RouterOS komunikuje na TCP (Transmission Control Protocol) portu 8291. Jedná se o jednoduchou grafickou nadstavbu, která umožňuje uživateli plně konfigurovat veškeré funkce RouterOS. Aplikace WinBox podporuje automatické vyhledávání zařízení s RouterOS v síti (tzv. Neighbor Viewer). Vyhledávání je kompatibilní jak s platformou x86, tak i se zařízeními RouterBoard. Přihlášení k zařízení je pak možné na základě jeho MAC adresy (obdoba MAC-Telnet) nebo IP adresy. WinBox je plně

kompatibilní s operačním systémem Microsoft Windows. V případě operačního systému Linux je nutné WinBox spouštět přes aplikaci Wine. Není zde však zaručena stoprocentní kompatibilita, což může vést k nestabilitě aplikace a k problémům během konfigurace síťového prvku [15].

Alternativou aplikace WinBox je webové rozhraní WebFig. WebFig je standardně součástí RouterOS a může být spuštěn přímo z webového prohlížeče zadáním IP adresy síťového prvku. WebFig umožňuje monitorování a plnohodnotnou konfiguraci RouterOS. Velkou výhodou webového rozhraní WebFig je nezávislost na platformě. Tato vlastnost umožňuje konfiguraci RouterOS z různorodých mobilních zařízení [14].

Poslední metodou je možnost komunikace s RouterOS pomoci MikroTik API (Application Programming Interface). Jedná se o aplikační rozhraní, které uživatelům umožňuje vytvářet vlastní softwarová řešení pro přímou konfiguraci RouterOS či pro získávání jeho konfiguračních informací. O MikroTik API je podrobněji pojednáno v následující sekci [10].

1.2 MikroTik API

Jak již bylo řečeno, MikroTik API umožňuje flexibilní způsob komunikace s RouterOS. Komunikace přes API standardně probíhá na TCP portu 8728 a je nešifrována. RouterOS verze 6.1 přinesla druhou variantu komunikace označenou jako MikroTik API-SSL. API-SSL využívá TCP port 8729 a zajišťuje šifrovanou komunikaci s RouterOS [4, 5].

Obě varianty jsou ve výchozím nastavení RouterOS aktivovány. Jejich stav lze zkontrolovat příkazem */ip service print*. Výstup příkazu je zobrazen na obr. 1.1.

```
[admin@MikroTik] > /ip service print
Flags: X - disabled, I - invalid
 #   NAME       PORT ADDRESS
 0 XI telnet      23
 1   ftp          21
 2   www          80
 3   ssh          22
 4 XI www-ssl     443
 5   api        8728
 6   winbox     8291
 7   api-ssl    8729
```

Obr. 1.1: Dostupné služby v RouterOS.

1.2.1 API-SSL

Služba API-SSL je schopna pracovat ve dvou režimech - s certifikátem a bez certifikátu. V případě režimu bez certifikátu se pro vytvoření spojení s RouterOS používá Diffie-Hellmanův protokol. Režim s certifikátem pak umožňuje vytvořit relaci s RouterOS pomocí protokolu TLS (Transport Layer Security). Režim s certifikátem bude využit v rámci implementace webové aplikace [11].

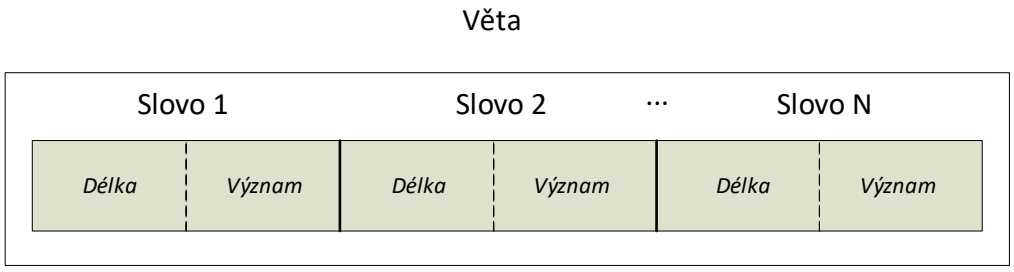
Aby bylo možné režim s certifikátem využívat, je nutné nejprve certifikát vygenerovat. Obecně lze certifikát vytvořit třemi způsoby [12]:

- na síťovém prvku s RouterOS – generaci certifikátu společně s privátním a veřejným klíčem lze provést přímo na síťovém prvku, čímž získáme certifikát podepsaný sám sebou. Takto vytvořený certifikát lze následně importovat na komunikující klientskou stanici.
- na klientské stanici přes OpenSSL – pomocí nástroje OpenSSL lze na klientské stanici vytvořit privátní klíč a certifikát společně s veřejným klíčem. Certifikát je opět podepsán sám sebou a je možné ho na síťovém prvku importovat.
- důvěryhodnou certifikační autoritou – klientská stanice zašle svůj veřejný klíč nezávislé certifikační autoritě. Certifikační autorita vytvoří certifikát, který následně podepíše svým privátním klíčem a zašle ho zpět klientské stanici. Klientská stanice poté zašle svůj certifikát na síťový prvek, který podpis certifikátu ověří svým veřejným klíčem. Pokud je podpis platný, je komunikace umožněna.

Po sestavení šifrovaného spojení mezi klientskou stanicí a síťovým prvkem již komunikace standardně probíhá dle služby API-SSL.

1.2.2 Princip komunikace přes službu MikroTik API

Komunikace přes službu MikroTik API probíhá na principu odesílání a přijímání vět. Struktura věty je zobrazena na obr. 1.2.



Obr. 1.2: Struktura věty u služby MikroTik API [10].

Klientská stanice zasílá jednotlivé věty na síťový prvek, přičemž každou získanou větu se síťový prvek snaží vyhodnotit a zpracovat. Pokud je přijatá věta úspěšně

zpracována, dojde k zaslání odpovědi síťového prvku zpět na klientskou stanici. V opačném případě je zaslána chybová hláška [10].

Každá věta se skládá ze sekvence slov, přičemž prázdné věty jsou síťovým prvkem ignorovány. Každé slovo pak obsahuje zakódovanou informaci obsahující délku slova. Délka slova vyjadřuje počet bitů, které budou odeslány. Za délkou slova již následuje zakódovaný význam slova. Konec každé věty je indikován slovem s nulovou délkou [10].

Význam slova lze obecně rozdělit do pěti částí [10]:

- **Příkaz (Command word)** – první slovo věty vždy obsahuje příkazové slovo (příkaz, který požadujeme na síťovém prvku vykonat). Příkaz vždy začíná znakem /, přičemž tento znak následně zastupuje mezery mezi zřetězeným příkazem v příkazovém řádku RouterOS. Příklad příkazů je zobrazen na obr. 1.3.

```
/login
/ip/dhcp-client/enable
/ip/address/add
```

Obr. 1.3: Příklad příkazů [10].

- **Atribut (Attribute word)** – za každým příkazovým slovem se nachází seznam atributů. Atributy se mohou za příkazovým slovem vkládat v libovolném pořadí. Každý atribut začíná znakem = společně se jménem atributu. Znak = zároveň slouží jako oddělovač pro nastavení konkrétní hodnoty atributu. Hodnota atributu může být nulová. Příklad atributů je zobrazen na obr. 1.4.

```
=name=admin
=numbers=1
=address=192.168.56.101=network=192.168.56.0=interface=ether2
```

Obr. 1.4: Příklad atributů [10].

- **API atribut (API attribute word)** – speciálním případem atributu je API atribut neboli tag. Pomocí tagu lze spustit více příkazů současně, aniž by bylo potřeba čekat na dokončení předchozího. Přijaté odpovědi síťového prvku jsou pak vždy označeny příslušným tagem, který byl zadán jako atribut příkazového slova. Tag vždy začíná syntaxí .tag a je nutné mu přiřadit unikátní číslo, které

označuje jednotlivé příkazy. Na obr. 1.5 je zobrazen příklad, jakým způsobem je možné tagy využívat.

```
/ip/dhcp-client/enable
=numbers=1
.tag=1
/ip/address/add
=address=192.168.56.101=network=192.168.56.0=interface=ether2
.tag=2
```

Obr. 1.5: Příklad použití tagů [10].

- **Dotaz (Query word)** – pro zjištění konfiguračních informací síťového prvku lze využít systém dotazů. Dotaz lze specifikovat pouze za příkazem print, přičemž každý dotaz začíná znakem ?. Příkaz print standardně vrací ke každé konfigurační položce unikátní identifikátor OID (Object Identifier), kterým lze jednotlivé položky odlišit. Speciálním atributem příkazu print je =.proplist=. Pomocí tohoto atributu lze filtrovat získané konfigurační položky. Úspěšnost dotazu je kontrolována na základě souboru booleovských hodnot. Pokud soubor hodnot obsahuje alespoň jednu nepravdu, tak je celý dotaz považován za chybný. Tato situace může nastat v případě, kdy je dotaz směřován na část konfigurace, která na síťovém prvku neexistuje nebo prozatím nebyla nakonfigurována. Dotaz pro zjištění, na kterém rozhraní běží DHCP (Dynamic Host Configuration Protocol) klient, je zobrazen na obr. 1.6.

```
/ip/dhcp-client/print
?status=bound
```

Obr. 1.6: Příklad dotazu [10].

- **Odpověď (Reply word)** – odpověď je zasílána pouze síťovým prvkem jako reakce na přijatou větu obdrženou od klientské stanice. Každá odpověď začíná znakem !. V případě přijetí bezchybného dotazu síťový prvek odpovídá označením !re, za kterým následují nakonfigurované atributy dotazované položky. Na vzniklé chyby během komunikace síťový prvek reaguje odpovědí !trap, za kterou se nachází vygenerovaná zpráva o chybě společně s kategorií, do které vzniklá chyba spadá. Odpověď síťového prvku je ukončena syntaxí !done. Pokud je komunikace přes službu MikroTik API přerušena, pak je generována

odpověď !fatal a následně je spojení mezi klientskou stanicí a síťovým prvkem ukončeno. Příklad odpovědi je znázorněn na obr. 1.7.

```
/system/package/print
!re
=.id=*5802
=name=system
=version=6.40.3
=scheduled=
!done
```

Obr. 1.7: Příklad odpovědi [10].

2 POUŽITÉ NÁSTROJE

Pro vytváření webové aplikace je primárně využit programovací jazyk Python společně s webovým frameworkem Django. V menší míře je pak zastoupen programovací jazyk JavaScript s využitím knihovny jQuery a technologií Ajax (Asynchronous JavaScript and XML). Manipulace s databází je prováděna přes relační databázový systém SQLite. Pro zobrazení webové aplikace je použit značkovací jazyk HTML (HyperText Markup Language) společně s kaskádovými styly CSS (Cascading Style Sheets). Pro implementaci monitorovacího systému do webové aplikace je využita otevřená platforma Grafana společně s databází InfluxDB a kolektorem Telegraf.

2.1 Python

Python je vysokoúrovňový, objektově orientovaný programovací jazyk s jednoduchou syntaxí. Je velmi vhodný především pro vytváření skriptů a webových aplikací. Python je hojně využíván i jako podpůrný programovací jazyk v rámci vývoje složitých softwarových řešení [8].

Python lze charakterizovat jako interpretovaný programovací jazyk. Znamená to, že vytvořený program není potřeba kompilovat, což má za následek značné urychlení produktivity během vývoje programu. Součástí jazyka Python je i standardní knihovna, která obsahuje velké množství modulů a balíčků pro import [8].

Syntaxe jazyka Python je oproti ostatním programovacím jazykům značně zjednodušená. Příkladem může být ukončení příkazu na řádku, kde není potřeba zadávat středník. Tím se výrazně sníží pravděpodobnost chyb způsobené opomenutím středníků na konci příkazů. K dalšímu zjednodušení lze zařadit úbytek závorek u bloků příkazů. Závorky jsou nahrazeny odsazením všech řádků patřících do konkrétního příkazového bloku. Jako další lze zmínit zjednodušený způsob definice proměnných. Během deklarace není totiž potřeba definovat jejich datový typ. Python sám rozpozná, o jaký datový typ proměnné se jedná [8].

2.2 Django

Django je otevřený webový framework založený na jazyce Python. Jeho hlavním cílem je umožnit snadnější a rychlejší vývoj interaktivních webových aplikací [4].

Základním stavebním kamenem při vývoji webové aplikace v Djangu jsou tzv. Django aplikace. Webová aplikace může být složena z jedné nebo několika těchto Django aplikací. Každá Django aplikace je pak složena z několika předem definovaných modulů napsaných v jazyce Python. Mezi tyto moduly patří [4]:

- migrations – složka, ve které se nacházejí moduly vygenerované po propojení webové aplikace s databází.
- ___init.py___ – prázdný modul, který slouží pro označení vytvořené Django aplikace. Aplikace je označena jako balíček Pythonu.
- admin.py – modul, který umožňuje zobrazit obsah databáze v rozhraní pro administrátora.
- apps.py – konfigurační modul pro vytvořenou Django aplikaci.
- models.py – databázový modul, ve kterém lze vytvořit databázovou tabulku. Tabulka je vytvořena pomocí třídy, přičemž sloupce tabulky se vytvoří pomocí atributů této třídy.
- views.py – modul, který obsahuje skupinu funkcí jazyka Python. Argumentem každé funkce je proměnná request, která přijímá požadavky uživatele v rámci webové stránky. Každá funkce pak vrací určitý obsah zpět na webovou stránku, který se uživateli zpětně zobrazí.
- urls.py – modul, který obsahuje schéma URL (Uniform Resource Locator) namapovaných na jednotlivé funkce v modulu views.py.

Během vývoje lze k webové aplikaci přistupovat přes webový prohlížeč zadáním lokální adresy 127.0.0.1 a portu 8000 [4].

2.3 SQLite

SQLite je knihovna funkcí sloužící ke správě relačních databází. Jedná se o odlehčenou variantu tradičních databázových systémů jako jsou např. MySQL nebo PostgreSQL. SQLite je zcela zdarma a může být libovolně používán [7].

SQLite pro svou funkci nevyžaduje samostatně bežící proces či server, neboť knihovna SQLite čte a zapisuje přímo do svých lokálně úložných souborů. Vzniklý soubor má obvykle příponu .db a může obsahovat aktuální data uložená ve více tabulkách. Soubor je nezávislý na platformě a je dobře přenositelný [7].

2.4 HTML, CSS a JavaScript

HTML je značkovací jazyk určený pro tvorbu webových stránek. Základem webové stránky jsou HTML elementy, které stanovují její strukturu. Každý HTML element je definován pomocí HTML tagů, které mohou obsahovat specifické atributy. Každý HTML tag je definován syntaxí <nazev_tagu></nazev_tagu>. Použitím HTML tagů je možné definovat záhlaví a tělo webové stránky, nadpisy, odstavce, odkazy a další. Jazyk HTML se obecně využívá i pro zapouzdření jiných skriptovacích jazyků [16].

Společně s jazykem HTML se velmi často využívají kaskádové styly CSS. Pomocí CSS je možné aplikovat různorodé styly na vytvářenou webovou stránku a měnit tak její vzhled. Styly CSS je možné definovat v záhlaví HTML kódu mezi tagy <style></style> nebo skrz importovaný konfigurační soubor s příponou .css [16].

Dynamickou funkcionalitu webové stránky zajišťuje programovací jazyk JavaScript. JavaScript je současně považován za skriptovací jazyk, který běží přímo v rámci webového prohlížeče. Příkazy JavaScriptu jsou definovány uvnitř HTML kódu mezi tagy <script></script> [16].

Součástí JavaScriptu je knihovna jQuery, která umožňuje podstatně zjednodušit psaní dynamických funkcí webové stránky. Mezi tyto funkce lze zařadit animace, efekty, řízení událostí a mnoho dalších. Knihovna jQuery je v práci použita především pro vytvoření komunikace mezi technologií Ajax a webovým serverem Django [16].

Technologie Ajax je soubor metod, které umožňují výměnu informací mezi webovým prohlížečem a webovým serverem. Tato výměna informací probíhá na pozadí a je uživateli skryta. Technologie Ajax je rovněž využita pro automatickou aktualizaci obsahu v rozhraní webové aplikace. Změna obsahu se ihned projeví a uživatel není tedy nucen rozhraní webové aplikace manuálně aktualizovat [16].

2.5 Grafana, InfluxDB a Telegraf

Grafana je otevřená platforma pro vizualizaci a následnou analýzu metrik. Pod pojmem metrika si lze představit veškeré získané měřitelné údaje popisující stav určitého systému. Pomocí Grafany lze vytvářet přehledné dashboardy, které lze libovolně upravovat a přizpůsobovat dle potřeb uživatele. Součástí dashboardu mohou být grafy, histogramy, heatmapy, tabulky a další. Grafana taktéž obsahuje již zabudovaný systém pro zasílání varovných zpráv neboli alerting. Alerting umožňuje v rámci monitorování nastavit určitou hranici hodnot neboli threshold. V případě překročení této hranice dojde k zaslání varovné zprávy přímo administrátorovi [5].

Grafana je obvykle nasazována na webové servery jako služba běžící na pozadí. Pro přístup na Grafanu lze využít webového prohlížeče zadáním IP adresy serveru společně s portem, na kterým Grafana naslouchá. Ve výchozím nastavení se jedná o port 3000 [3].

Základní funkčnost grafany spočívá v zasílání dotazů na databázi neboli zdroj dat. V databázi jsou poté uloženy veškeré metriky, které Grafana postupně zpracovává. Grafana v současné době podporuje celkem 8 databází. Mezi tyto databáze patří Graphite, InfluxDB, CloudWatch, ElasticSearch a další. V rámci řešení práce

byla zvolena databáze InfluxDB z důvodu přehledné dokumentace a jednoduché instalace [6].

Posledním nástrojem v rámci monitorování je kolektor. Kolektor slouží pro periodickou kolekci a ukládání metrik do databáze. V rámci práce byl zvolen kolektor zvaný Telegraf, který umožňuje periodicky zasílat SNMP (Simple Network Management Protocol) dotazy na síťové prvky. Telegraf byl zvolen z důvodu přehledných a dobře editovatelných konfiguračních souborů a taktéž z důvodu zajištěné kompatibility s databází InfluxDB [17].

3 VÝVOJ WEBOVÉ APLIKACE

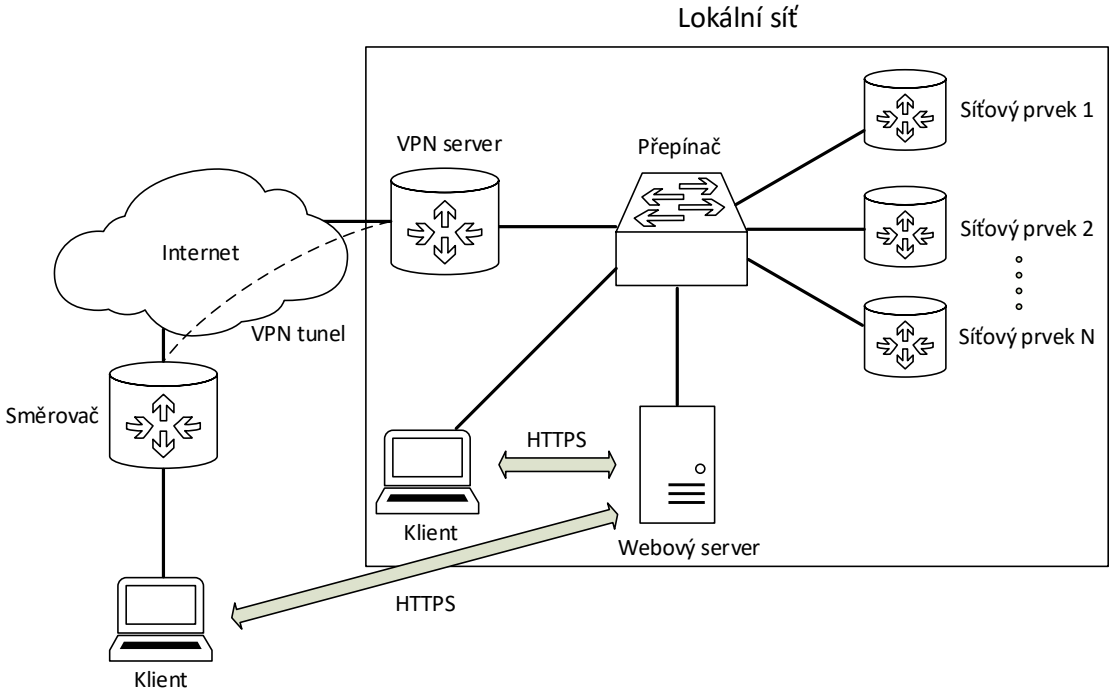
Vývoj webové aplikace je rozdělen do dvou částí. První část vývoje je zaměřena na implementaci řešení pro síťové prvky, které se nacházejí v továrním nastavení. Druhá část vývoje již řeší síťové prvky připravené pro komunikaci přes službu API-SSL.

3.1 Síťové prvky v továrním nastavení

Aby bylo možné službu API-SSL používat, musí být splněny dva základní požadavky:

- **Zajištění základní konektivity síťových prvků** – síťové prvky v továrním nastavení nemají nastavenou IP adresu na žádném rozhraní. Není tedy možné se na síťový prvek připojit přes síťovou vrstvu ISO/OSI modelu. Zároveň není možné službu API-SSL využít, neboť pro její běh je zapotřebí IP adresa daného síťového prvku. Je tedy potřeba problém s IP adresou vyřešit a zajistit tak spojení se síťovým prvkem na síťové vrstvě.
- **Generace a import certifikátu** – síťové prvky v továrním nastavení rovněž neobsahují žádné importované certifikáty, které služba API-SSL vyžaduje. Je tedy nutné zajistit generaci a import certifikátu na daný síťový prvek.

Pro hromadnou správu síťových prvků přes webovou aplikaci byla sestavena síťová topologie, jejíž schéma se nachází na obr. 3.1.

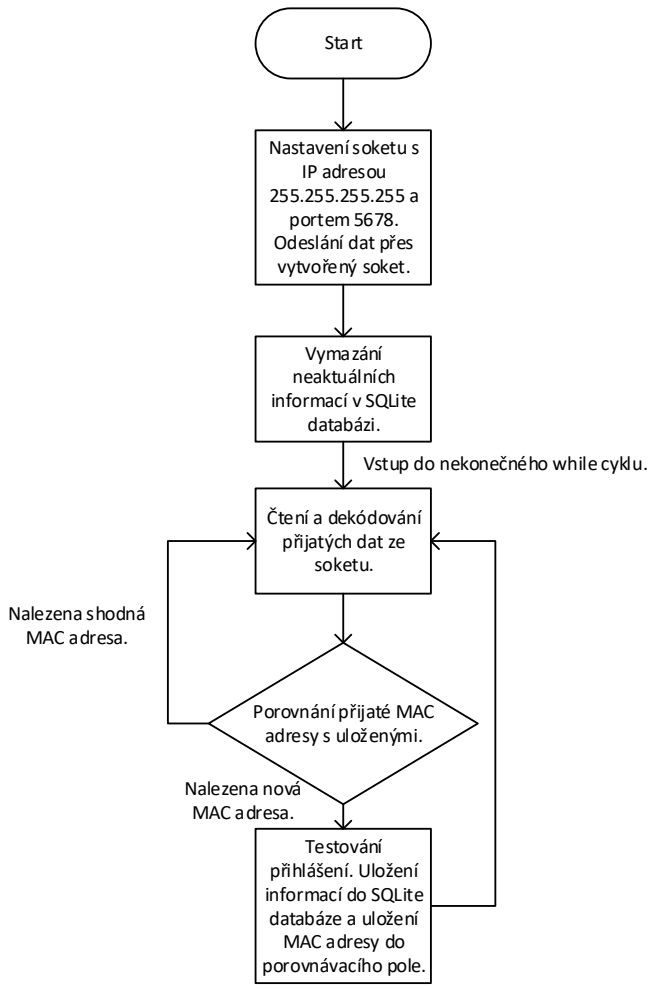


Obr. 3.1: Schéma síťové topologie.

Topologie je složena z klientské stanice, webového serveru, přepínače, N síťových prvků a VPN serveru. Klientská stanice komunikuje s webovým serverem přes protokol HTTPS (HyperText Transfer Protocol Secure) zadáním IP adresy webového serveru. Webový server, na kterém běží webová aplikace, je připojen k portu přepínače. K ostatním portům přepínače jsou pak připojeny jednotlivé síťové prvky MikroTik s operačním systémem RouterOS. Aby bylo možné na webový server přistupovat i z vnější sítě, je do portu přepínače zapojen i VPN (Virtual Private Network) server. Klient, který se nachází ve vnější síti, je po sestavení zabezpečeného VPN tunelu schopen s webovým serverem komunikovat tak, jako kdyby se nacházel uvnitř lokální sítě.

3.1.1 Nalezení síťových prvků

Zapojené síťové prvky je nejprve potřeba v lokální síti nalézt. Pro tento účel byl vytvořen skript, jehož vývojový diagram je zobrazen na obr. 3.2.



Obr. 3.2: Vývojový diagram skriptu pro nalezení síťových prvků.

Funkce skriptu spočívá v opakovaném vysílání paketů na všesměrovou IP adresu 255.255.255.255 a UDP (User Datagram Protocol) port 5678. Na portu 5678 standardně naslouchá protokol MNDP (MikroTik Neighbor Discovery Protokol), který se nachází na každém síťovém prvku s RouterOS a je aktivní již v továrním nastavení. Protokol MNDP slouží k poskytnutí základních informací dotazovaného síťového prvku. Odchozí pakety adresovány na všesměrovou IP adresu 255.255.255.255 jsou po zapouzdření do rámce adresovány na všesměrovou MAC adresu FF:FF:FF:FF:FF:FF. Použitím všesměrové MAC adresy je zajištěno, že MNDP dotaz dorazí na každý síťový prvek v lokální síti, neboť jednotlivé síťové prvky jsou součástí jedné všesměrové domény.

MNDP dotaz je opakovaně zasílán webovou aplikací umístěnou na webovém serveru. Tím je dosaženo, že webová aplikace je schopná nalézt nově připojené síťové prvky. Odpovědi síťových prvků jsou pak zasílány zpět na MAC adresu webového serveru. Je zde však nutné ošetřit, aby nedocházelo k získávání duplicitních informací síťových prvků obdržených v předchozí iteraci. Ve skriptu je tedy vytvořena podmínka, která kontroluje nově přijaté MAC adresy síťových prvků s již naučenými. Pokud se přijatá MAC adresa rovná naučené, je ignorována. Hexadecimální data přijatých paketů jsou zároveň skriptem dekódována. Ze získaných dat síťových prvků je pak vybrána MAC adresa, IP adresa, rozhraní pro správu, identita a verze RouterOS.

Během každé iterace je zároveň spouštěna funkce, která testuje přihlášení na každý nalezený síťový prvek. Testování je automaticky prováděno pomocí protokolu MAC-Telnet zadáním výchozích přihlašovacích údajů síťového prvku. Zda bylo přihlášení úspěšné či nikoliv je rozpoznáno prostřednictvím dekódovaných hexadecimálních dat, které síťový prvek odeslal zpět. Pokud se funkci podaří na síťový prvek přihlásit, pak vrátí textový řetězec OK. V opačném případě vrátí textový řetězec FAILED. Následně jsou nalezené síťové prvky společně se získanými údaji ukládány do databáze SQLite. Princip ukládání síťových prvků do databáze je popsán v následující kapitole.

Spouštění skriptu je zajištěno pomocí funkce *subprocess.call()*, která je volána v modulu views.py. Pomocí knihovny threading bylo dosaženo, že proces dotazování probíhá na pozadí a nikterak neovlivňuje běh webové aplikace. Zároveň je v kódu vyřešen problém s hromaděním nově vytvořených procesů v důsledku opakovaného spouštění skriptu webovou aplikací. Při každém novém spuštění skriptu totiž dojde k restartování stávajícího procesu. Nemůže tedy dojít k situaci, kde by se při každém dalším spuštění skriptu generoval nový proces.

3.1.2 Princip ukládání síťových prvků do databáze

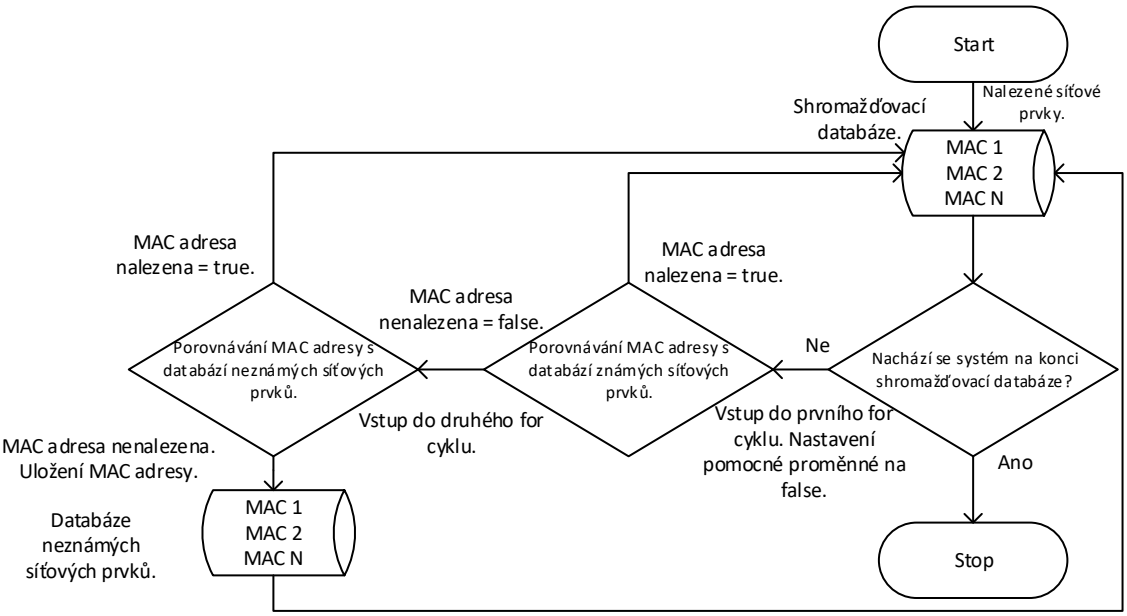
Pro přehledné ukládání a třídění síťových prvků byl vytvořen systém, který se skládá ze tří separátních databází:

- databáze známých síťových prvků – jedná se o databázi, kde se nacházejí síťové prvky, které byly uživatelem označeny jako známé. V této databázi je navíc vytvořen sloupec, který označuje skupinu, do které síťový prvek patří. Uživatel tedy může skrz webovou aplikaci třídit známé sítové prvky do patřičných skupin. V rámci webové aplikace je implementována funkce třídění síťových prvků do třech skupin. Jedná se o skupinu studentských síťových prvků, skupinu směrovačů a skupinu přepínačů.
- databáze neznámých síťových prvků – v této databázi se nacházejí síťové prvky, které nebyly prozatím uživatelem roztříděny do patřičných skupin. Zároveň se mohou v této databázi nalézat i takové síťové prvky, které byly nově připojeny v rámci lokální sítě.
- shromažďovací databáze – tato databáze slouží jako dočasné úložiště nově nalezených síťových prvků. Jedná se v podstatě o mezistupeň během procesu ukládání. Při každém novém spuštění vyhledávacího skriptu je databáze kompletně vymazána a nalezené síťové prvky jsou postupně do databáze ukládány znovu. Tato databáze byla vytvořena za účelem správného rozdělování nalezených síťových prvků mezi obě předchozí databáze. Druhým důvodem k vytvoření této databáze je kontrola zapojení síťových prvků v topologii. Pokud dojde k odpojení síťového prvku, dojde zároveň i k vymazání jeho záznamu z databáze.

Během každé iterace vyhledávání síťových prvků dojde k inicializaci výše uvedených databází. Následně jsou veškeré nalezené síťové prvky ukládány do shromažďovací databáze. Každý řádek databáze pak reprezentuje samostatný síťový prvek. Jako unikátní identifikátor každého uloženého síťového prvku v databázi byla zvolena jeho MAC adresa. V dalším kroku dojde ke spuštění prvního for cyklu, který postupně porovnává řádky shromažďovací databáze s řádky databáze známých síťových prvků. Všechny řádky jsou porovnávány na základě MAC adresy, která se nachází v prvním sloupci každého řádku databáze.

Porovnávání obou databází je založeno na pomocné proměnné typu boolean, která je ve výchozím stavu nastavena na false. Pokud je během porovnávání nalezena stejná MAC adresa, znamená to, že síťový prvek již existuje v databází známých síťových prvků. Při nalezení stejné MAC adresy dojde zároveň k uložení ostatních údajů ze shromažďovací databáze. Tímto je zajištěna aktualizace údajů všech známých síťových prvků. Současně dojde k nastavení pomocné proměnné na true a k přesunu na další řádek shromažďovací databáze.

V případě, kdy MAC adresa není v databázi známých zařízení nalezena, je pomocná proměnná ponechána na false a dojde ke spuštění druhého cyklu. Druhý cyklus porovnává řádky shromažďovací databáze s databází neznámých zařízení. Pokud během porovnávání není nalezena shodná MAC adresa, pak dojde k uložení síťového prvku do databáze neznámých zařízení a k přesunu na další řádek shromažďovací databáze. V opačném případě dojde pouze k přesunu na další řádek. Pokud se systém ukládání nachází na konci shromažďovací databáze, dojde k jeho ukončení. Na obr. 3.3 je znázorněn kompletní vývojový diagram systému pro ukládání nalezených síťových prvků.



Obr. 3.3: Vývojový diagram systému pro ukládání síťových prvků.

3.1.3 Implementace protokolu MAC-Telnet

Jak již bylo řečeno, síťové prvky v továrním nastavení neobsahují nakonfigurovanou IP adresu na žádném svém rozhraní. Není proto možné se na síťový prvek připojit na úrovni síťové vrstvy. K řešení tohoto problému je zvolen způsob připojení na linkové vrstvě pomocí MAC adresy síťového prvku. Pro tento způsob připojení je využita open source knihovna MAC-Telnet napsaná v jazyce Python.

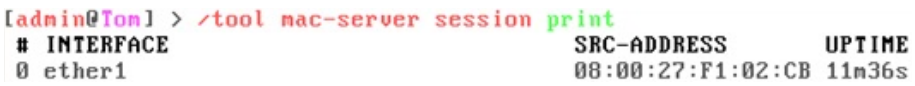
Princip komunikace přes protokol MAC-Telnet probíhá na základě modelu klient-server. Webová aplikace, ve které je implementována knihovna MAC-Telnet, plní funkci klienta (MAC-Telnet klient). Jednotlivé síťové prvky pak plní funkci serverů (MAC-Telnet server). Protokol MAC-Telnet standardně naslouchá na UDP portu 20561 jak na straně klienta, tak i na straně serveru.

Aby bylo možné se na síťový prvek připojit prostřednictvím knihovny MAC-Telnet, je nutné doplnit 2 parametry pro navázání spojení. Prvním parametrem je určení směru, ve kterém bude komunikace se síťovým prvkem probíhat. Prakticky se jedná o název rozhraní síťové karty webového serveru, který lze zjistit pomocí příkazu ifconfig v terminálovém okně. V případě webového serveru se jedná o rozhraní s označením eth0.

Druhým parametrem je MAC adresa síťového prvku, se kterým je potřeba navázat spojení. MAC adresu síťového prvku je již možné získat z údajů uložených v databázi a lze ji bez problému použít jako druhý parametr knihovny MAC-Telnet. Zadáním požadovaných parametrů dojde k navázání spojení se síťovým prvkem.

Komunikace se síťovým prvkem přes knihovnu MAC-Telnet probíhá na principu zasílání paketů na všesměrovou IP adresu 255.255.255.255. Nastává zde však problém při zapouzdření paketu do rámce. Po zapouzdření paketu se všesměrovou IP adresou je standardně získán rámec se všesměrovou MAC adresou. Je tedy nutné zaměnit všesměrovou MAC adresu za MAC adresu síťového prvku, se kterým je zapotřebí komunikovat. Tato MAC adresa je již vložena jako druhý parametr knihovny MAC-Telnet. Funkce knihovny pak zároveň přetváří původně získaný rámec na rámec s cílovou MAC adresou, která byla zadána v parametru.

Komunikace se síťovým prvkem je udržována v cyklu, ve kterém lze zasílat příkazy na síťový prvek. Jednotlivé příkazy jsou funkcí kódovány do hexadecimálního tvaru a zároveň vkládány do vytvořených rámců. V cyklu jsou rovněž zasílány kontrolní rámce, které zajišťují, aby relace nebyla ze strany síťového prvku přerušena. Probíhající relaci lze na síťovém prvku zkontrolovat příkazem */tool mac-server session print*. Výstup příkazu lze vidět na obr. 3.4.



Obr. 3.4: Relace MAC-Telnet na síťovém prvku.

Na obrázku lze vidět, na kterém rozhraní síťového prvku relace probíhá a jak dlouho. Výstupem příkazu je rovněž MAC adresa webového serveru, se kterým síťový prvek momentálně komunikuje.

3.1.4 Zajištění konektivity na síťové vrstvě

Zajištění konektivity na síťové vrstvě je řešeno na základě přiřazení IP adresy na rozhraní síťového prvku. Obecně se jedná o rozhraní, které je v lokální síti propojeno s rozhraním přepínače. Rozhraní s nakonfigurovanou IP adresou lze označit jako

rozhraní pro správu. IP adresu lze na rozhraní síťového prvku nakonfigurovat dvěma způsoby:

- staticky – jedná se o manuální konfiguraci IP adresy na rozhraní. Společně s IP adresou je taktéž nutné manuálně nakonfigurovat odpovídající síťovou masku. IP adresa sítě je pak automaticky dopočítána síťovým prvkem dle přiřazené síťové masky.
- dynamicky – jedná se o dynamické přiřazení IP adresy pomocí DHCP (Dynamic Host Configuration Protokol) klienta na rozhraní síťového prvku. DHCP klient je standardně na síťovém prvku s RouterOS k dispozici. Ve výchozím nastavení však není aktivní na žádném rozhraní. Aby bylo možné získat IP adresu pomocí DHCP klienta, musí se v lokální síti nacházet DHCP server s nakonfigurovaným rozsahem IP adres.

Dynamická konfigurace

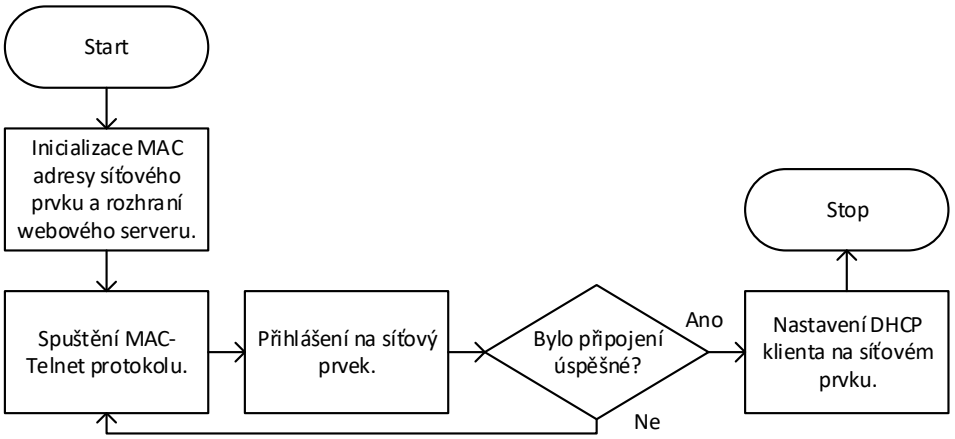
Pro aktivaci DHCP klienta na síťovém prvku byl vytvořen samostatný skript, který je webovou aplikaci volán přes funkci umístěnou v modulu views.py. Dočasné připojení na síťový prvek je zajištěno přes MAC-Telnet protokol, který je ve skriptu implementován pomocí knihovny MAC-Telnet.

Aby bylo možné DHCP klienta aktivovat, je zapotřebí přes MAC-Telnet protokol provést určitou sekvenci příkazů. Ve skriptu je sekvence příkazů vytvořena pomocí modulu pexpect. Pexpect pracuje na principu očekávání konkrétního výstupu po odeslání příkazu na zařízení, se kterým komunikace probíhá. V případě síťového prvku s RouterOS je po připojení přes MAC-Telnet protokol nejdříve očekáván výstup pro přihlášení. Přihlašovací údaje, které jsou definovány ve skriptu, jsou modulem pexpect zaslány zpět na síťový prvek. Po úspěšném přihlášení je očekáván znak >, který označuje, že je síťový prvek připraven přijímat příkazy.

Předposledním krokem je zadání příkazu pro spuštění DHCP klienta. Přes modul pexpect je odeslán příkaz */ip dhcp-client add interface=name disabled=no*, kde name je proměnná obsahující název rozhraní, na kterém je požadováno DHCP klienta aktivovat. Po odeslání příkazu je dále nutné opět definovat očekávaný znak >, jinak by došlo k chybě a předchozí příkaz by se neprovedl. Celá sekvence příkazu je zakončena funkcí, která korektně přeruší relaci se síťovým prvkem.

Během řešení tohoto skriptu se vyskytl problém, kdy po zadání přihlašovacích údajů docházelo k nepravidelným výpadkům spojení se síťovým prvkem. Tento jev pravděpodobně souvisí s nestabilitou MAC-Telnet protokolu, která je způsobena neúplnou kompatibilitou mezi síťovým prvkem a MAC-Telnet knihovnou. Pro řešení tohoto problému je ve skriptu implementován cyklus, ve kterém skript během připojování neustále testuje, zda došlo k úspěšnému připojení či nikoliv. V případě

neúspěchu dojde k opakovanému připojení. V opačném případě se provede požadovaná sekvence příkazů a cyklus se ukončí. Vývojový diagram skriptu se nachází na na obr. 3.5.



Obr. 3.5: Vývojový diagram skriptu pro spuštění DHCP klienta na síťovém prvku.

Po ukončení skriptu je na požadovaném rozhraní síťového prvku aktivován DHCP klient, který automaticky přiřadí příslušnou IP adresu získanou z adresního rozsahu DHCP serveru. Po přidělení IP adresy je dále možné řešit problematiku zajištění certifikátu na síťovém prvku.

3.1.5 Generace a import certifikátu

Generace certifikátu je prováděna na webovém serveru pomocí nástroje OpenSSL. Webový server vytvoří vlastnoručně podepsaný certifikát, který je možné na jednotlivé síťové prvky importovat. Zároveň je webový server v rámci lokální sítě samostatná certifikační autorita.

Po dokončení generování nástrojem OpenSSL je získán klíč server.key a certifikát server.crt. Oba získané soubory je zapotřebí nahrát na síťový prvek. K tomuto účelu byly vytvořeny 2 funkce, které jsou dále využívány jako součást dalších skriptů sloužících ke konfiguraci síťových prvků.

První funkce slouží k nahrání klíče a certifikátu webového serveru na síťový prvek. Přenos klíče a certifikátu je realizován pomocí knihovny Paramiko, která zajišťuje implementaci protokolu SSH pro jazyk Python. Konkrétně je pro přenos využit protokol SFTP (Secure File Transfer Protocol), který je součástí knihovny Paramiko. Přesněji se jedná o implementaci protokolu FTP (File Transfer Protocol) nad protokolem SSH. Aby bylo možné přenést klíč a certifikát přes SFTP, musí být tedy na síťovém prvku aktivován protokol SSH. Protokol SSH je u síťových prvků MikroTik obvykle standardně aktivován již v továrním nastavení.

Knihovna Paramiko pracuje na principu komunikace klient-server. Při inicializaci funkce je nejprve vytvořen SSH objekt, který reprezentuje SSH klienta. V tomto případě je SSH klientem webový server a SSH serverem je síťový prvek MikroTik. V dalším kroku dojde k pokusu o připojení na SSH server, které je realizováno prostřednictvím IP adresy síťového prvku společně s portem 22 a přihlašovacími údaji síťového prvku. Tímto způsobem dojde k vytvoření SSH tunelu, kde je veškerá komunikace šifrována. Na závěr dojde k otevření protokolu SFTP uvnitř vytvořeného SSH tunelu a k následnému přenosu klíče a certifikátu na síťový prvek.

Druhou funkcí je import klíče a certifikátu na daném síťovém prvku a následně přiřazení certifikátu ke službě API-SSL. Import certifikátu a klíče do systému RouterOS je prováděn přímo z jeho souborového systému, do kterého byl certifikát nahrán protokolem SFTP. Pro komunikaci se síťovým prvkem je využita knihovna Tikapy napsaná v jazyce Python, která obsahuje implementaci aplikačních rozhraní MikroTik API a API-SSL. Jednotlivé příkazy a atributy služeb API i API-SSL jsou v rámci knihovny Tikapy definovány jako textové řetězce umístěné v proměnné datového typu list.

Prostřednictvím MikroTik API dojde k přihlášení na síťový prvek zadáním jeho IP adresy a portu 8728. Následně je funkcí poslána sekvence příkazů, které umožní službu API-SSL dále využívat. Import certifikátu je proveden zadáním příkazu */certificate import, =file-name=server.crt*. Obdobně pak import klíče příkazem */certificate import, =file-name=server.key*. Přiřazení certifikátu ke službě API-SSL je provedeno příkazem */ip service set api-ssl, =certificate=server.cer_0*. Na obr. 3.6 je zobrazen výstup příkazu */ip service print* po aktivaci certifikátu u služby API-SSL.

```
[admin@MikroTik] > /ip service print
Flags: X - disabled, I - invalid
 #   NAME      PORT ADDRESS                                 CERTIFICATE
 0 XI telnet     23
 1   ftp         21
 2   www         80
 3   ssh         22
 4 XI www-ssl    443                                        none
 5   api       8728
 6   winbox    8291
 7   api-ssl   8729                                         server.crt_0
```

Obr. 3.6: Přiřazený certifikát u služby API-SSL.

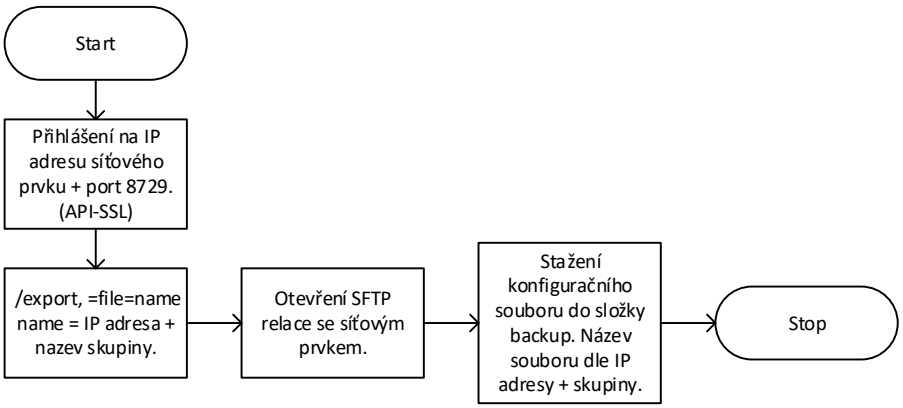
3.2 Síťové prvky komunikující na síťové vrstvě

Síťové prvky s nakonfigurovanou IP adresou a přiřazeným certifikátem již mohou komunikaci přes API-SSL plně využívat. V rámci postupu dalšího řešení bylo potřeba implementovat významné nástroje pro správu síťových prvků. Mezi tyto nástroje patří záloha a import konfiguračních souborů, možnost konfigurační soubory

editovat, automatické vykreslování síťové topologie a možnost monitorování stavu síťových prvků. V poslední řadě bylo potřeba zajistit, aby byl zachován charakter hromadné správy s ohledem na co nejjednodušší ovladatelnost webové aplikace.

3.2.1 Záloha konfiguračních souborů

Záloha aktuálních konfiguračních souborů síťových prvků je vytvořena pomocí skriptu, jehož vývojový diagram je znázorněn na obr. 3.7.



Obr. 3.7: Vývojový diagram skriptu pro zálohu konfiguračních souborů.

Tento skript se skládá ze dvou funkcí. První funkce slouží k vytvoření konfiguračního souboru přímo v souborovém systému síťového prvku. Parametrem funkce je IP adresa síťového prvku a název skupiny, do které síťový prvek patří. Před zahájením komunikace dojde nejprve k připojení na síťový prvek pomocí API-SSL a k automatickému přihlášení. Následně je přes API-SSL vyslán příkaz k vytvoření konfiguračního souboru */export, =file=name*, kde proměnná name označuje, s jakým názvem bude soubor vytvořen. Název souboru je složen z IP adresy síťového prvku a jeho skupiny. Tímto je zajištěna situace, ve které uživatel pozná, k jakému síťovému prvku konfigurační soubor patří. Konfigurační soubor je ukládán ve formátu rsc. Zda byl konfigurační soubor úspěšně vytvořen lze na síťovém prvku zkontrolovat pomocí příkazu */file print*, jak je znázorněno na na obr. 3.8.

```
[admin@MikroTik] > /file print
 # NAME                  TYPE                        SIZE CREATION-TIME
 0 192.168.1.185Router... script                     264 mar/18/2018 13:04:48
```

Obr. 3.8: Vytvořený konfigurační soubor v souborovém systému síťového prvku.

Druhá funkce skriptu slouží ke stažení vytvořeného konfiguračního souboru ze síťového prvku přímo na webový server. Tímto způsobem dojde k vytvoření aktuální zálohy konfiguračního souboru. Parametrem funkce je rovněž IP adresa síťového

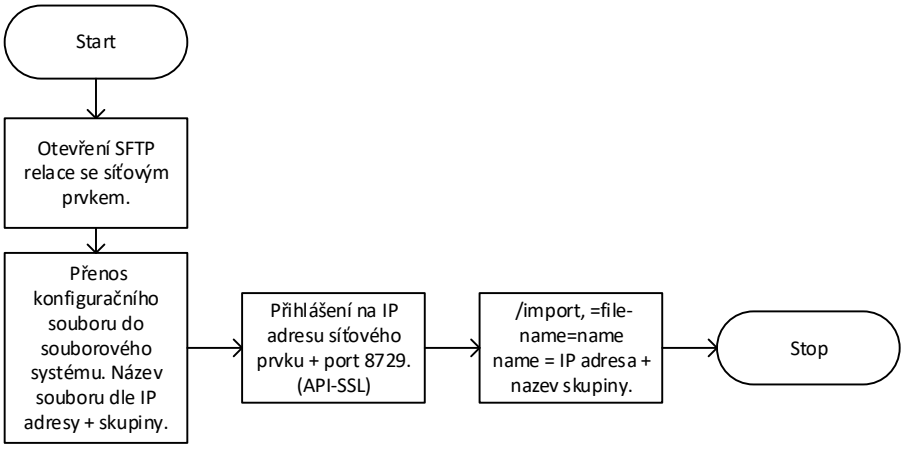
prvku a jeho skupina. Stažení konfiguračního souboru je realizováno prostřednictvím protokolu SFTP knihovny Paramiko. Jedná se o stejný princip implementace, jako při přenosu certifikátu a klíče na síťový prvek. Stažený konfigurační soubor je standardně na webovém serveru uložen do složky backups. Konfigurační soubor je rovněž uložen ve formátu rsc a pojmenován dle IP adresy síťového prvku a jeho skupiny.

3.2.2 Resetování konfigurace

Pro resetování konfigurace vybraného síťového prvku byl vytvořen jednoduchý skript. Přes službu API-SSL je zaslán příkaz */system/reset-configuration, =no-defaults=yes* na vybranou IP adresu síťového prvku. Po zaslání tohoto příkazu dojde k navrácení všech provedených změn, které byly na síťovém prvku provedeny. Atribut *=no-defaults=yes* zajistí, aby nedošlo k načtení žádné výchozí konfigurace. Zároveň dojde k restartování vybraného síťového prvku.

3.2.3 Import výchozí konfigurace

Pro import výchozí konfigurace na síťové prvky byl vytvořen další skript, který pracuje se třemi výchozími konfiguračními soubory. Každý konfigurační soubor je určen pro konkrétní skupinu, do které síťový prvek patří. Jedná se o skupinu studentských síťových prvků, skupinu směrovačů a skupinu přepínačů. Konfigurační soubory jsou pro přehlednost rovněž pojmenovány dle názvů skupin. Všechny tři konfigurační soubory jsou uloženy ve formátu rsc na webovém serveru. Vývojový diagram skriptu je zobrazen na obr. 3.9.



Obr. 3.9: Vývojový diagram skriptu pro import konfiguračních souborů.

Princip skriptu opět spočívá ve využití dvou funkcí. První funkce slouží k nahrání výchozího konfiguračního souboru z webového serveru na síťový prvek. Parametrem

funkce je IP adresa síťového prvku a název skupiny. K přenosu konfiguračního souboru z webového serveru na síťový prvek opět slouží protokol SFTP. Konfigurační soubor je následně uložen do souborového systému síťového prvku ve formátu rsc a se jménem příslušné skupiny.

Druhá funkce skriptu slouží k samotnému importu výchozího konfiguračního souboru přímo na síťovém prvku. Během importu dojde k přepsání aktuální konfigurace síťového prvku konfigurací obsaženou ve výchozím konfiguračním souboru. Komunikace se síťovým prvkem opět probíhá přes API-SSL. Nejprve dojde k automatickému přihlášení na síťový prvek. V případě úspěšného přihlášení je dále vyslán příkaz */import, =file-name=name*, kde proměnná name je název výchozího konfiguračního souboru uloženého v souborovém systému síťového prvku.

3.2.4 Editor výchozích konfigurací

Pro flexibilní manipulaci výchozích konfiguračních souborů byl do webové aplikace implementován editor, přes který lze konfigurační soubory síťových prvků libovolně upravovat. Implementace editoru je založena na třech textových polích, kde každé pole patří pro danou skupinu síťových prvků. Implementace editoru do webové aplikace je založena na základě editace konfiguračních souborů, které jsou uložené na webovém serveru. Jedná se o soubory student.rsc, router.rsc a switch.rsc.

Pro zobrazení aktuálního stavu všech konfiguračních souborů v textových polích byla v modulu views.py vytvořena samostatná funkce. Princip funkce spočívá v otevření a přečtení uloženého konfiguračního souboru pro danou skupinu. Obsah souboru je následně uložen do proměnné a převeden do formátu JSON (JavaScript Object Notation). Formát JSON se zde nachází v podobě klíč - objekt, kde objekt reprezentuje celý obsah konfiguračního souboru. Formát JSON je zde využit pro zajištění výměny dat během volání této funkce webovou aplikací. Volání funkce je zajištěno přes technologii Ajax zasláním HTTP metody GET na URL této funkce. Metoda GET je zasílána vždy při načtení editoru webové aplikace. Toto chování vždy zajistí zobrazení aktuálního stavu konfiguračních souborů ve všech textových polích.

Editace a ukládání změn v konfiguračních souborech probíhá téměř inverzně k výše uvedenému postupu. Nejprve je pomocí jazyku JavaScript uložen celý obsah textového pole do proměnné a následně převeden do formátu JSON. Dále je přes technologii Ajax pomocí HTTP metody POST tento obsah zaslán na URL další funkce vytvořenou v souboru views.py. Tato funkce získaný obsah textového souboru uloží do své proměnné a převede ho z formátu JSON na prostý text datového typu string. Následně je funkcí otevřen příslušný konfigurační soubor a veškerý text je do něj zapsán.

Poslední schopností editoru je možnost nahrát změněný konfigurační soubor na síťové prvky patřící k dané skupině. Pro tento účel byla vytvořena poslední funkce v souboru views.py, která je opět volána HTTP metodou POST pomocí technologie Ajax. Nejdříve je inicializována databáze známých síťových prvků. Následně jsou pomocí for cyklu procházeny všechny řádky databáze. Během procházení řádků dochází zároveň ke kontrole skupin nalezených síťových prvků. Pokud síťový prvek patří do příslušné skupiny, dojde k zavolání funkce pro import výchozí konfigurace.

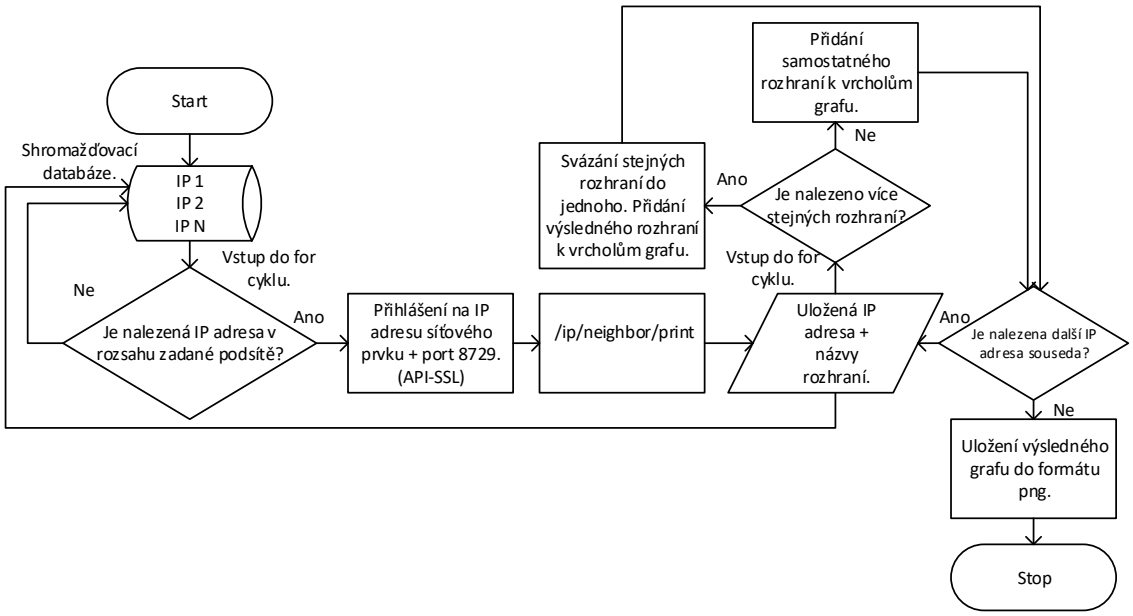
3.2.5 Vykreslování síťové topologie

Vykreslování síťové topologie je implementováno prostřednictvím dalšího skriptu. Princip skriptu spočívá v postupném dotazování jednotlivých síťových prvků na jejich sousedy. Nejprve je ve skriptu inicializována shromažďovací databáze. Důvodem inicializace shromažďovací databáze je zajištění vykreslování síťové topologie pro známé i neznámé síťové prvky současně. Následně jsou z této databáze vyhledávány IP adresy jednotlivých síťových prvků. Vyhledávání je prováděno pomocí for cyklu, ve kterém je zároveň nastavena podmínka, která kontroluje, zda jsou nalezené IP adresy součástí stejné podsítě. V rámci urychlení vykreslování síťové topologie je vyhledávání omezeno na podsíť 192.168.0.0/16, což odpovídá rozsahu od 192.168.0.1 do 192.168.255.254 v rámci použitelných IP adres této podsítě.

Pro každou nalezenou IP adresu síťového prvku je následně spuštěna funkce pro nalezení sousedících síťových prvků. Princip funkce spočívá v zaslání dotazu */ip/neighbor/print* přes službu API-SSL na nalezenou IP adresu síťového prvku, která je zároveň parametrem této funkce. Výstup příkazu je ve formátu klíč - hodnota, kde klíč reprezentuje sousedící síťový prvek a hodnota reprezentuje jeho parametry. Z těchto parametrů je následně vybrána IP adresa sousedícího síťového prvku a název rozhraní, se kterým je sousedící síťový prvek propojen. Oba tyto parametry jsou uloženy do svých proměnných a svázány s nalezenou IP adresou síťového prvku.

Pomocí modulu NetworkX a Matplotlib je následně ze získaných údajů vykreslen graf síťové topologie. Celý proces vykreslování je implementován pomocí jediné funkce. Nejprve je funkcí inicializován prázdný graf. Pomocí for cyklu jsou dále postupně doplňovány vrcholy a hrany grafu, přičemž vrcholy grafu jsou označeny dle IP adresy síťových prvků. Názvy rozhraní jsou zde použity z důvodu rozhodování, zda je síťový prvek připojen do přepínače či nikoliv. Pokud je zkoumaný síťový prvek připojen do přepínače, pak může nastat situace, ve které příkaz */ip/neighbor/print* vrátí více sousedů napojených na stejné rozhraní zkoumaného síťového prvku. Pro tento případ je v cyklu vytvořena podmínka, která více stejných rozhraní sváže do jednoho a jako souseda zkoumaného síťového prvku určí přepínač. Přepínač je pak přidán jako samostatný vrchol grafu.

Na závěr je celý graf uložen ve formátu png na webový server a pomocí jazyku Javascript nahrán do rozhraní webové aplikace. Vývojový diagram skriptu je zobrazen na obr. 3.10.



Obr. 3.10: Vývojový diagram skriptu pro vykreslování síťové topologie.

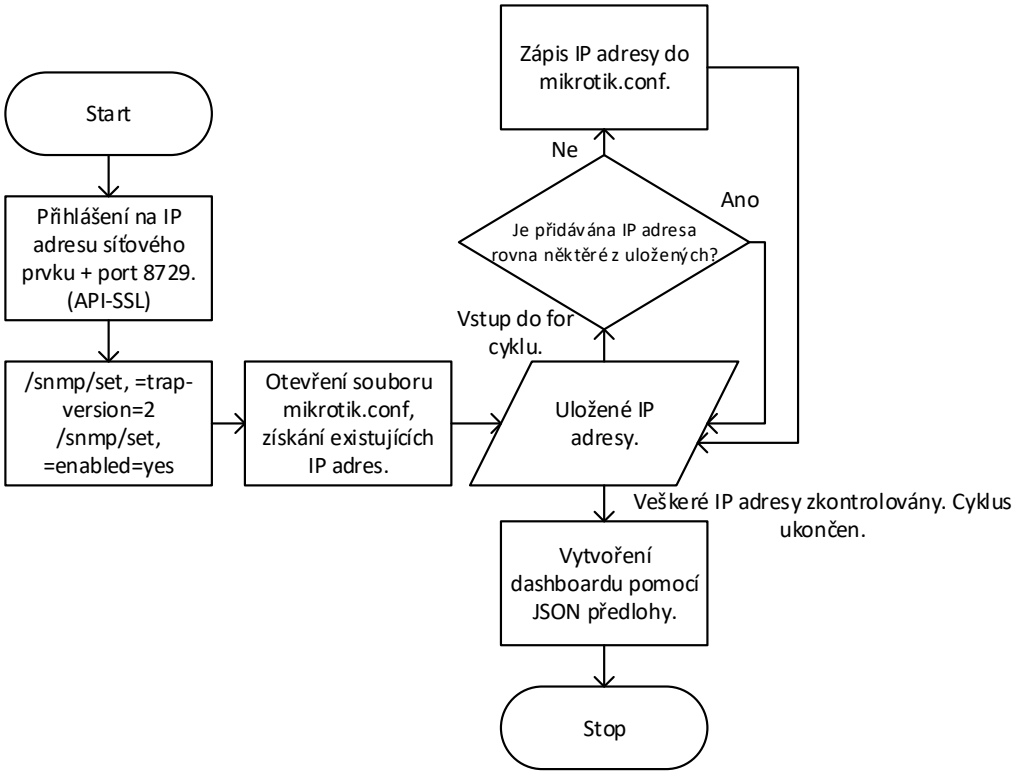
3.2.6 Monitorovácí systém

Monitorování síťových prvků je realizováno prostřednictvím rozhraní z nástroje Grafana, které je implementováno do prostředí webové aplikace. Rozhraní se skládá z jednotlivých dashboardů, kde každý dashboard patří pro jeden monitorovaný síťový prvek. Jednotlivé dahsboardy jsou pro přehlednost pojmenovány dle IP adresy monitorovaného síťového prvku. Každý dashboard obsahuje celkem tři grafy:

- graf vytíženosti procesoru – sleduje stav vytíženosti procesoru síťového prvku.
- graf využité paměti – sleduje množství využité paměti síťového prvku.
- graf provozu na rozhraní – sleduje příchozí a odchozí provoz všech aktivních rozhraní síťového prvku.

Do monitorovacího systému je implementována možnost přidání či odebrání monitorovaného síťového prvku. Síťové prvky MikroTik je možné monitorovat prostřednictvím zasílání SNMP dotazů na vybrané identifikátory OID. Identifikátory OID poskytují SNMP protokolu informace o stavu určitého systému v rámci síťového prvku. Síťové prvky MikroTik mají standardně zakázáno přijímat jakékoliv zprávy zasílané SNMP protokolem. Pokud na síťový prvek dorazí jakýkoliv SNMP dotaz, pak dojde automaticky k jeho zahození. Tato situace je ve skriptu ošetřena prostřednictvím funkce, která umožňuje zapnout přijímání SNMP dotazů na síťovém prvku.

Tato funkce je zároveň prvním krokem, který se provede během přidání síťového prvku do monitorovacího systému. Pro přidání síťového prvku do monitorovacího systému byl vytvořen skript, jehož vývojový diagram je znázorněn na obr. 3.11.



Obr. 3.11: Vývojový diagram skriptu pro přidání zařízení do monitorovacího systému.

Princip skriptu spočívá v zaslání dvou příkazů přes službu API-SSL na síťový prvek. Prvním z nich je příkaz */snmp/set, =trap-version=2*, který nastaví verzi SNMP protokolu na síťovém prvku. Pro jednodušší implementaci byla zvolena verze 2. Druhým příkazem je příkaz */snmp/set, =enabled=yes*, který aktivuje přijímání SNMP dotazů.

Aby bylo možné síťové prvky monitorovat hromadně a zároveň v pravidelných intervalech, je dále nutné správně nastavit SNMP kolektor. Jak již bylo řečeno, pro monitorovací systém webové aplikace byl zvolen SNMP kolektor zvaný Telegraf. Pro nastavení Telegrafu byl vytvořen separátní konfigurační soubor pojmenovaný jako mikrotik.conf umístěný ve složce /etc/telegraf/telegraf.d. Uvnitř konfiguračního souboru je nutné nejprve specifikovat seznam agentů, kterým bude Telegraf zasílat SNMP dotazy. V případě monitorovacího systému webové aplikace je agentem IP adresa síťového prvku. Dalším krokem je nastavení intervalu, ve kterém bude Telegraf opakovaně zasílat SNMP dotazy na síťový prvek. S ohledem na vytížení sítě a síťových prvků byl zvolen interval odesílání na jednu minutu. Nejdůležitější

části konfiguračního souboru jsou pak samotná OID, na které Telegraf zasílá SNMP dotazy. V konfiguračním souboru byly specifikovány OID pro vytížení procesoru, množství využité paměti a pro příchozí a odchozí provoz na rozhraní.

Druhá funkce skriptu tedy slouží k automatickému nastavení agentů výše uvedeného konfiguračního souboru mikrotik.conf. Argumentem funkce je IP adresa přidaného síťového prvku. Nejprve je funkcí otevřen samotný soubor mikrotik.conf, ze kterého jsou za pomocí regulárních výrazů získány IP adresy již přidaných síťových prvků. IP adresy jsou zároveň ukládány do lokální proměnné typu list. Za pomocí for cyklu je dále porovnávána IP adresa nově přidávaného síťového prvku s IP adresami již přidaných síťových prvků. Pokud se IP adresa nově přidaného síťového prvku nerovná žádné již existující IP adrese, je tato IP adresa do konfiguračního souboru přidána. V opačném případě je IP adresa přeskočena a do konfiguračního souboru zapsána není. Tato podmínka byla implementována z toho důvodu, aby nedocházelo k duplicitnímu zápisu stejných IP adres do konfiguračního souboru.

Tímto způsobem je zajištěn automatický zápis IP adresy do konfiguračního souboru kolektoru Telegraf. Telegraf je na webovém serveru spuštěn jako služba na pozadí. Při každé změně konfiguračního souboru je potřeba službu restartovat. Pro tuto situaci byla na webovém serveru vytvořena úloha v plánovači úloh s názvem Cron. Cron v intervalu jedné minuty vždy Telegraf restartuje.

Posledním krokem k přidání nového síťového prvku do monitorovacího systému je vytvoření nového dashboardu v nástroji Grafana. Pro tento účel je vytvořena třetí funkce skriptu, která umožňuje vytvořit nový dashboard speciálně pro nově přidaný síťový prvek. Dashboard je vytvořený na základě IP adresy síťového prvku, která je zároveň argumentem funkce. Komunikace s Grafanou je založena na základě HTTP API, což je samostatné aplikační rozhraní Grafany. HTTP API pracuje na principu zasíláni HTTP metod GET a POST. Metoda GET slouží k získání dat ze zdroje, respektive ze specifikovaného URL Grafany. Metoda POST pak slouží k zaslání dat na URL Grafany. Použití obou metod je ve skriptu realizováno prostřednictvím modulu requests.

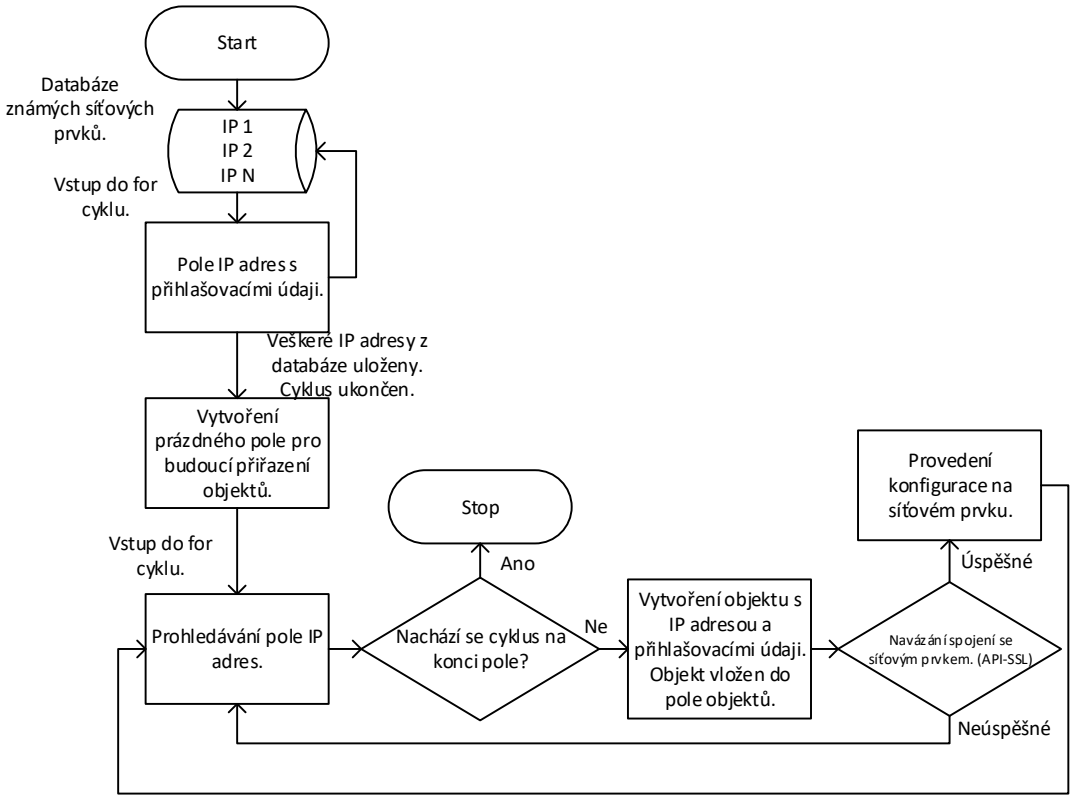
Pro nový dashboard přidaného síťového prvku byla vytvořena předloha, která je ve skriptu implementována ve formátu JSON. Pomocí metody POST je pak předloha ve formátu JSON poslána skrz HTTP API na URL Grafany. Konkrétně se jedná o URL */api/dashboards/db*. Při úspěšném vytvoření dashboardu je vrácen identifikátor UID (Unique Identifier), který je funkcí uložen do databáze SQLite k danému síťovému prvku.

Odebrání síťového prvku z monitorovacího systému probíhá téměř inverzně k výše uvedenému postupu. Nejprve je přes službu API-SSL odeslán příkaz */snmp/-set, =enabled=no*, který zakáže přijímat SNMP dotazy na vybraném síťovém prvku. Poté dojde k odebrání IP adresy z pole agentů v kolektoru Telegraf. Na závěr je

skrz HTTP API vyslán příkaz k odstranění dashboardu daného síťového prvku z nástroje Grafana. Odstranění dashboardu probíhá na základě identifikátoru UID, který je svázán s daným síťovým prvkem v databázi SQLite.

3.2.7 Princip hromadné správy síťových prvků

Pro zajištění hromadné správy síťových prvků byly navrhnuty a implementovány dva samostatné algoritmy. Oba algoritmy byly během vývoje webové aplikace střídavě využívány v závislosti na momentálně řešeném problému. Vývojový diagram prvního algoritmu se nachází na obr. 3.12



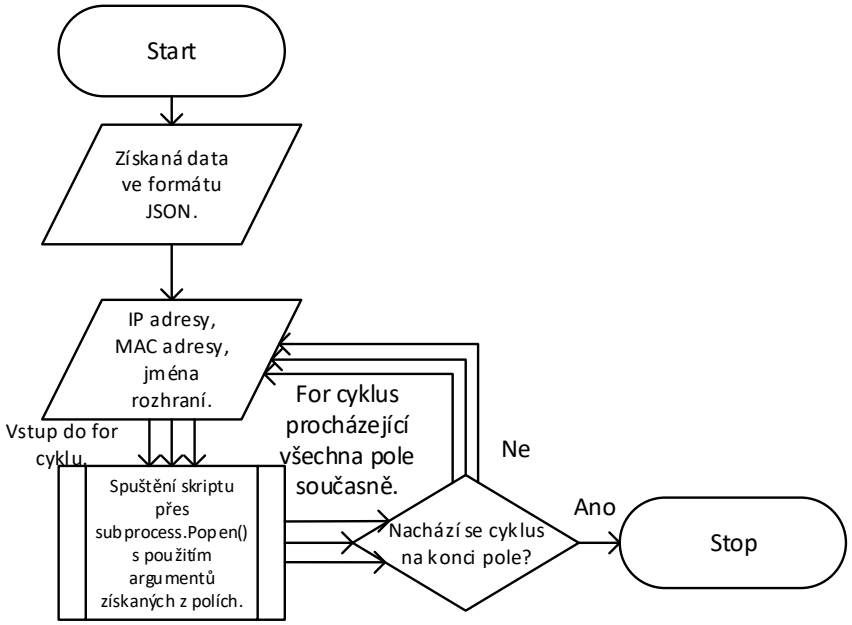
Obr. 3.12: Vývojový diagram prvního algoritmu pro hromadnou správu síťových prvků.

Princip algoritmu spočívá ve vytvoření pole objektů, kde jednotlivé objekty reprezentují síťové prvky v lokální síti. Předpisem objektů je třída, která obsahuje metody pro přihlášení a konfiguraci síťových prvků. Objekty jsou z třídy dynamicky vytvářeny v závislosti na IP adresách a přihlašovacích údajích jednotlivých síťových prvků. IP adresy jsou pomocí for cyklu získávány z databáze známých síťových prvků a postupně zapisovány do samostatného pole. Tímto způsobem je zajištěno, aby algoritmus vždy vytvořil určitý počet objektů v závislosti na počtu IP adres

síťových prvků v poli. Přihlašovací údaje jsou nastaveny na výchozí a jsou svázány s každou IP adresou. Pokud je například požadováno konfigurovat tři síťové prvky najednou, dojde k vytvoření tří objektů na základě třech IP adres v poli.

V dalším kroku je potřeba zajistit, aby došlo k postupnému připojení a následné konfiguraci jednotlivých síťových prvků nacházejících se v poli objektů. Tento problém je jednoduše vyřešen na základě for cyklu, který postupně prochází pole objektů. V těle cyklu jsou pak volány požadované metody třídy. Zároveň je v cyklu pomocí výjimky ošetřena situace neúspěšného přihlášení na síťový prvek. Pokud se algoritmu nepodaří na síťový prvek přihlásit, dojde k jeho přeskočení a k výběru následující IP adresy z pole společně s vytvořením příslušného objektu.

Druhý algoritmus zajišťuje stejnou funkci hromadné správy jako výše uvedený. Rozdílem je však mírně odlišná implementace. Vývojový diagram druhého algoritmu je zobrazen na obr. 3.13.



Obr. 3.13: Vývojový diagram druhého algoritmu pro hromadnou správu síťových prvků.

Na počátku algoritmu dojde k uložení získaných dat do příslušných polí. Veškerá data jsou algoritmem získána z odpovědí, které zasílá webová aplikace prostřednictvím technologie Ajax ve formátu JSON. Jaká data algoritmus získá závisí na vykonané události webové aplikace. Může se jednat o IP adresy, MAC adresy či jména rozhraní síťových prvků. Pomocí for cyklu jsou pak jednotlivá pole současně procházena. Při každé iteraci for cyklu je pomocí funkce *subprocess.Popen* spuštěn skript, který provede požadovanou akci na příslušném síťovém prvku.

Všechny skripty, které jsou v tomto algoritmu spouštěny, vyžadují přiřazení ar-

gumentu. Argumentem jsou skriptu předána data z jednotlivých polí, se kterými skript následně pracuje. Uvnitř skriptů je nastavena sekvence funkcí, které se při každé iteraci spouštějí za sebou. Výchozí přihlašovací údaje jsou rovněž definovány uvnitř každého skriptu. Tímto způsobem je zajištěna obsluha všech požadovaných síťových prvků.

3.2.8 Ověření komunikace přes API a API-SSL

Vývoj webové aplikace rovněž zahrnoval ověření komunikace se síťovými prvky. Pomoci síťového analyzátoru Wireshark byla provedena kontrola komunikace přes běžné API a jeho šifrovanou variantu API-SSL. Na obr. 3.14 a na obr. 3.15 se nachází výstup ze síťového analyzátoru Wireshark.

```
192.168.56.101  192.168.56.102  TCP     8728 > 36444 [PSH, ACK] Seq=46 Ack=74 Win=14496 Len=7 TSval=471867 TSecr=11142267
192.168.56.102  192.168.56.101  TCP     36444 > 8728 [PSH, ACK] Seq=74 Ack=53 Win=29312 Len=1 TSval=11142289 TSecr=471867
192.168.56.101  192.168.56.102  TCP     8728 > 36444 [ACK] Seq=53 Ack=75 Win=14496 Len=0 TSval=471871 TSecr=11142289
192.168.56.102  192.168.56.101  TCP     36444 > 8728 [PSH, ACK] Seq=75 Ack=53 Win=29312 Len=61 TSval=11142327 TSecr=471871

0000  08 00 27 84 4a 5e 08 00  27 f1 02 cb 08 00 45 00   ..'.J^.. '.....E.
0010  00 71 90 ba 40 00 40 06  b7 b0 c0 a8 38 66 c0 a8   .q..@.@. ....8f..
0020  38 65 8e 5c 22 18 23 6a  94 cb cf 18 21 06 80 18   8e.\".#j ....!...
0030  00 e5 f2 7f 00 00 01 01  08 0a 00 aa 04 b7 00 07   ........ ........
0040  33 3f 2f 69 70 2f 61 64  64 72 65 73 73 2f 61 64   3?/ip/ad dress/ad
0050  64 1a 3d 61 64 64 72 65  73 73 3d 31 39 32 2e 31   d.=addre ss=192.1
0060  36 38 2e 35 36 2e 31 30  34 2f 32 34 11 3d 69 6e   68.56.10 4/24.=in
0070  74 65 72 66 61 63 65 3d  65 74 68 65 72 33 00      terface= ether3.
```

Obr. 3.14: Komunikace přes běžné API.

```
192.168.56.101  192.168.56.102  TCP     8729 > 33276 [PSH, ACK] Seq=2808 Ack=1464 Win=18784 Len=44 TSval=413770 TSecr=10561218
192.168.56.102  192.168.56.101  TCP     33276 > 8729 [PSH, ACK] Seq=1464 Ack=2852 Win=37888 Len=30 TSval=10561261 TSecr=413770
192.168.56.101  192.168.56.102  TCP     8729 > 33276 [ACK] Seq=2852 Ack=1494 Win=18784 Len=0 TSval=413770 TSecr=10561261
192.168.56.102  192.168.56.101  TCP     33276 > 8729 [PSH, ACK] Seq=1494 Ack=2852 Win=37888 Len=76 TSval=10561264 TSecr=413770

0000  08 00 27 84 4a 5e 08 00  27 f1 02 cb 08 00 45 00   ..'.J^.. '.....E.
0010  00 80 85 41 40 00 40 06  c3 1a c0 a8 38 66 c0 a8   ...A@.@. ....8f..
0020  38 65 81 fc 22 19 67 de  a7 8d 67 67 98 b0 80 18   8e..".g. ..gg....
0030  01 28 f2 8e 00 00 01 01  08 0a 00 a1 26 f0 00 06   .(...... ....&...
0040  50 4a 17 03 03 00 29 f3  3d 7e aa ec a2 a4 aa 1f   PJ....). =~......
0050  ea 64 71 e3 36 09 86 87  f6 1b ff 8f 5d aa 59 62   .dq.6... ....].Yb
0060  77 02 02 13 1c cb e6 6e  7d 2c 38 fd 3f 79 36 75   w......n },8.?y6u
0070  17 03 03 00 19 f3 3d 7e  aa ec a2 a4 ab 5c e3 8c   ......=~ .....\..
0080  8f 1b cd 3b 46 d7 f8 52  f1 5d 80 dd 35 10         ...;F..R .]..5.
```

Obr. 3.15: Komunikace přes API-SSL.

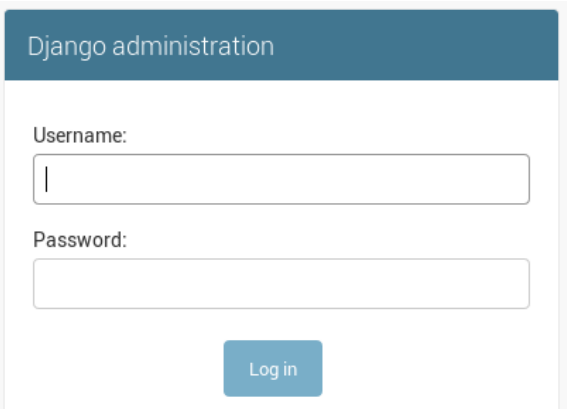
Na obou výstupech je analyzován paket pro nastavení IP adresy na rozhraní síťového prvku. Porovnáním obou výstupů lze vidět, že komunikace přes běžné API je nešifrována a lze ji bez problému odposlechnout. Naopak komunikace přes API-SSL je kompletně šifrována a lze ji považovat za plně funkční.

4 ROZHRANÍ WEBOVÉ APLIKACE

Webová aplikace je umístěna na webovém serveru s operačním systémem Ubuntu verze 16.04 LTS. Pro nasazení webové aplikace na webový server byla využita služba Apache verze 2.4.18. Nasazení webové aplikace na webový server vyžadovalo úpravu konfiguračního souboru služby Apache. Konkrétně se jedná o soubor default-ssl.conf umístěného ve složce /etc/apache2/sites-avaliable/. Uvnitř konfiguračního souboru bylo nutné službě Apache namapovat cestu ke složce, ve které je umístěno rozhraní WSGI (Web Server Gateway Interface). WSGI je součástí frameworku Django a slouží ke komunikaci mezi službou Apache a vytvořenou webovou aplikací. Na závěr bylo v konfiguračním souboru nastaveno, aby služba Apache automaticky přesměrovala veškerou komunikaci přes protokol HTTPS. Přístup do webové aplikace je zajištěn prostřednictvím webového prohlížeče zadáním IP adresy webového serveru.

4.1 Správa uživatelských účtů

V rámci webové aplikace lze ke správě uživatelských účtů přistupovat přes administrátorské prostředí frameworku Django. Do prostředí lze vstoupit zadáním URL do webového prohlížeče ve formátu IP adresa webového serveru/admin. Po zadání URL je uživatel vyzván k zadání administrátorského jména a hesla, jak lze vidět na obr. 4.1.



Obr. 4.1: Přihlášení do administrátorského prostředí.

Administrátorské účty lze vytvářet přímo na webovém serveru. K tomuto účelu slouží modul manage.py, který je umístěn ve složce webové aplikace. Zadáním příkazu *python manage.py createsuperuser* a následným vyplněním přihlašovacího jména a hesla dojde k vytvoření administrátorského účtu.

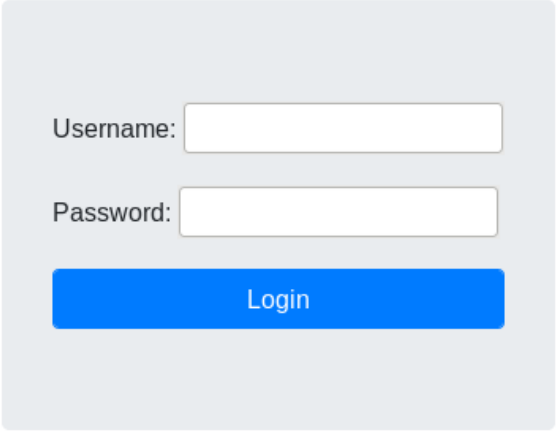
V administrátorském prostředí je možné libovolně vytvářet, mazat či upravovat uživatelské účty pro přístup do prostředí webové aplikace. Veškeré uživatelské

účty jsou uloženy v databázi Users. Pro vytvoření nového uživatelského účtu slouží tlačítko ADD USER. Po vyplnění přihlašovacího jména a hesla je uživatelský účet uložen do databáze. Přihlašovací jméno uživatele je v databázi uloženo v prostém textu. Heslo je naopak uloženo ve formě hashe a je prakticky nemožné ho zpětně z hashe zjistit. Pro ukládání hesel do databáze je využita kryptografická funkce PBKDF2 (Password-Based Key Derivation Function 2) společně s hashovací funkcí SHA-256 (Secure Hash Algorithm - 256).

Vytvořenému uživatelskému účtu lze libovolně měnit práva na přihlášení do webové aplikace či rovnou do administrátorského prostředí. Uživatelský účet lze rovněž deaktivovat nebo kompletně odstranit.

4.2 Příhlášení do systému

Přihlášení uživatele do prostředí webové aplikace je realizováno prostřednictvím autentizačního systému frameworku Django. Přihlášení je implementováno prostřednictvím funkce login, která je importována v modulu urls.py. Přihlašovací stránka webové aplikace je zobrazena na obr. 4.2.



Obr. 4.2: Přihlašovací stránka webové aplikace.

Přihlášení pracuje na principu porovnání vložených přihlašovacích údajů s údaji, které se nacházejí v databázi Users. Vložené přihlašovací jméno je porovnáváno přímo s přihlašovacím jménem uloženého v databázi. Z vloženého hesla je nejprve vypočítán hash, který se následně porovnává s uloženým hashem. Pokud přihlašovací údaje souhlasí, dojde k vytvoření samostatné relace a k přesměrování na domovskou stránku webové aplikace. Vytvořená relace zajišťuje, aby se uživatel nemusel přihlašovat na každé URL webové aplikace zvlášť. Relace je udržována po celou dobu přihlášení uživatele.

Odhlášení uživatele je implementováno pomocí funkce logout, která je importována opět v modulu urls.py. Odhlášení uživatele ze systému je realizováno kompletním ukončením relace. Po odhlášení je uživatel automaticky přesměrován na přihlašovací stránku webové aplikace.

4.3 Prostředí webové aplikace

Po úspěšném přihlášení je uživatel přesměrován do prostředí webové aplikace. Webová aplikace obsahuje celkem čtyři rozhraní, mezi kterými lze libovolně přepínat v menu v horní liště. Jednotlivá rozhraní jsou podrobněji popsána v následujících podkapitolách.

4.3.1 Monitoring

Rozhraní s názvem Monitoring slouží jako domovská stránka, na kterou je uživatel přesměrován ihned po přihlášení do webové aplikace. Celé rozhraní je zobrazeno na obr. 4.3.

Obr. 4.3: Rozhraní Monitoring.

Hlavním cílem tohoto rozhraní je uživateli zobrazit stav všech monitorovaných síťových prvků. K tomuto účelu byl do rozhraní implementován monitorovací systém z nástroje Grafana. Monitorovací systém obsahuje dashboardy pro monitorované síťové prvky. Mezi dashboardy lze přepínat pomocí vysouvacího menu v levé horní části monitorovacího systému. Každý dashboard je pojmenovaný dle IP adresy síťového prvku. Součástí dashboardu jsou tři grafy zobrazující vytížení procesoru,

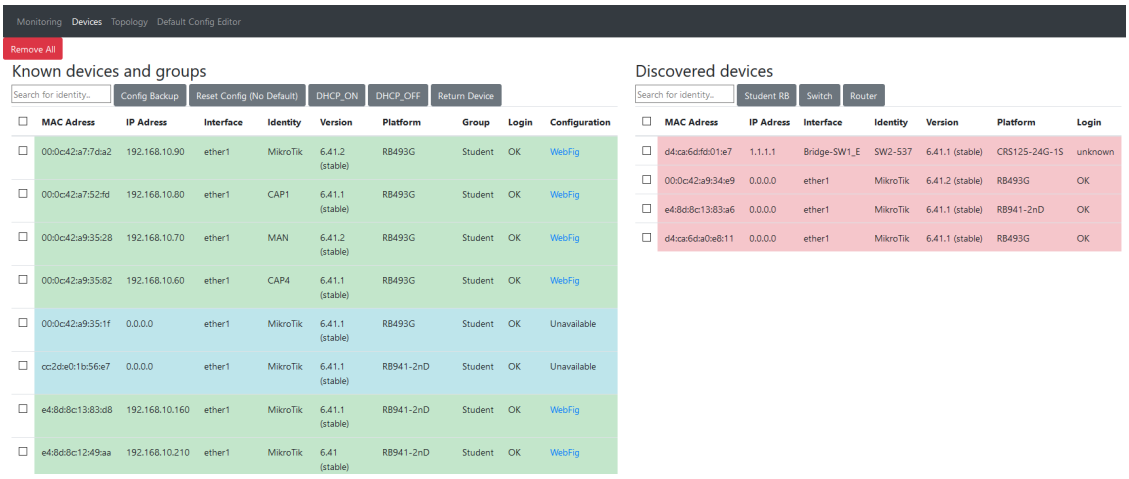
využití paměti a stav síťového provozu na všech aktivních rozhraních síťového prvku. V pravé horní části monitorovacího systému lze nastavit časové rozmezí všech grafů a interval, podle kterého se budou grafy současně aktualizovat. Pro zajištění bezpečnosti je celý monitorovací systém zpřístupněn pouze pro čtení a není tak možné grafy modifikovat.

V pravé části rozhraní se nachází seznam všech známých síťových prvků. Tento seznam zároveň umožňuje hromadně povolit či zakázat monitorování vybraných síťových prvků. Pomocí zaškrtávacích políček lze vybrat jednotlivé síťové prvky. Pomocí tlačítek Enable Monitoring a Disable Monitoring lze povolit či zakázat monitorování vybraných síťových prvků. Při povolení monitorování automaticky dojde k vytvoření nových dashboardů v monitorovacím systému. Počet nově vytvořených dashboardů odpovídá počtu vybraných síťových prvků.

Grafy nově vytvořených dashboardů jsou na počátku prázdné a neobsahují žádná data. Doba načtení dat do nově vytvořených grafů trvá přibližně jednu minutu. Důvodem je pravidelné restartování kolektoru Telegraf, které probíhá na pozadí přes plánovač úloh Cron. Po restartování kolektoru dojde automaticky k periodickému monitorování všech nově přidaných síťových prvků.

4.3.2 Devices

Rozhraní Devices poskytuje uživateli přehled všech zapojených síťových prvků v lokální síti. Rozhraní je zobrazeno na obr. 4.4.



Obr. 4.4: Rozhraní Devices.

Součástí rozhraní jsou dvě tabulky, které obsahují základní informace o připojených síťových prvcích. Tabulka na levé straně obsahuje seznam všech známých síťových prvků. Tabulka na pravé straně obsahuje nově nalezené síťové prvky, které doposud nebyly zařazeny mezi známé. Obě tabulky obsahují filtr, který umožňuje

vyfiltrovat síťové prvky na základě jejich názvů. Nově nalezené síťové prvky lze pomocí zaškrtávacích polí a tlačítek Student RB, Switch a Router roztřídit do příslušných skupin. Tímto způsobem dojde k přesunu síťových prvků do tabulky v levé části rozhraní. Pomocí tlačítka Return Device lze síťový prvek vrátit zpět mezi neznámé.

Známé síťové prvky jsou dále v tabulce rozlišeny barevně. Zelená barva označuje nakonfigurované síťové prvky, které mají nastavenou IP adresu na rozhraní a je možné s nimi komunikovat přes službu API-SSL. Modrá barva označuje zařízení, která nejsou nakonfigurována. Pomocí tlačitek DHCP_ON a DHCP_OFF lze zapnout či vypnout DHCP klienta na vybraných síťových prvcích. Zapnutí DHCP klienta zajistí automatické přiřazení IP adresy na připojeném rozhraní síťového prvku. V tabulce se přiřazení IP adresy projeví změnou barvy síťového prvku z modré na zelenou. Ve sloupci IP address lze vidět nově přiřazená IP adresa DHCP serverem.

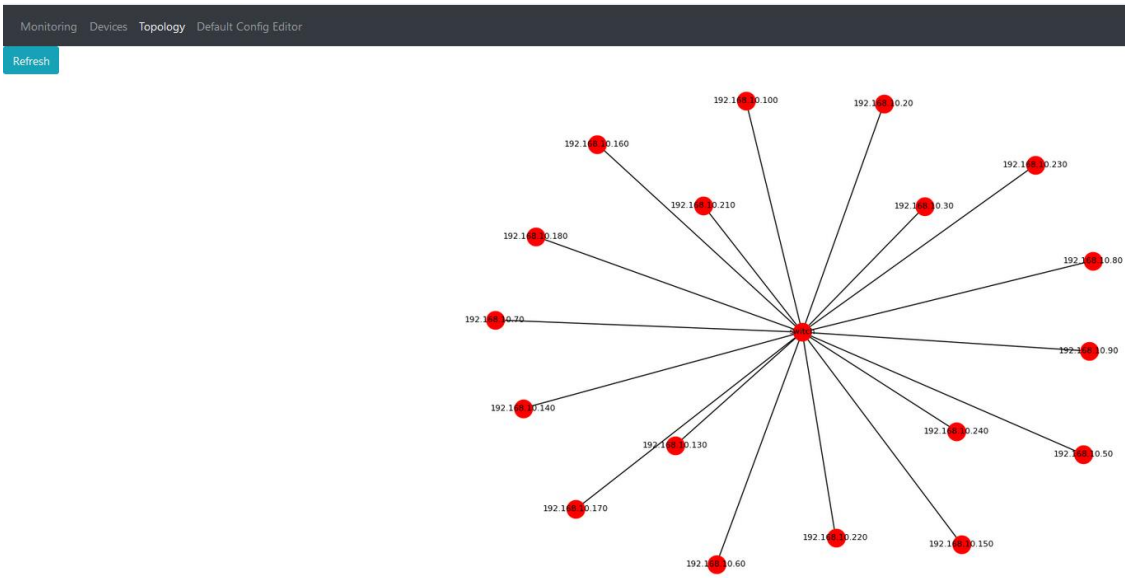
Součástí rozhraní jsou tlačítka, která slouží pro manipulaci s konfiguračními soubory síťových prvků. Tlačitko Config Backup umožňuje provést hromadnou zálohu konfiguračních souborů vybraných síťových prvků. Jednotlivé konfigurační soubory jsou ukládány na webový server do složky backups, která se nachází v adresáři webové aplikace. Tlačítko Reset Config (No Default) umožňuje hromadně vrátit veškeré konfigurační změny na označených síťových prvcích. Zároveň dojde k restartování všech označených síťových prvků.

Součástí tabulky známých síťových prvků je možnost přístupu na webové rozhraní WebFig. Po kliknutí na odkaz WebFig dojde k automatickému přesměrování do konfiguračního prostředí síťového prvku. Toto prostředí umožňuje uživateli provádět podrobnější konfigurace. Webové rozhraní WebFig je přístupné pouze na síťových prvcích, které mají nakonfigurovanou IP adresu na rozhraní. Pokud jsou na síťovém prvku nastaveny výchozí přihlašovací údaje, je uživatel do konfiguračního systému vpuštěn bez nutnosti přihlášení. V opačném případě je nutné zadat přihlašovací údaje síťového prvku. Poslední tlačítko s názvem Remove All umožňuje odstranit síťové prvky z obou tabulek současně a vynutit nové vyhledávání síťových prvků.

4.3.3 Topology

Rozhraní Topology umožňuje vykreslit diagram zapojení síťových prvků a poskytnout tak uživateli představu, jak jsou síťové prvky fyzicky propojeny. Tlačítkem Refresh dojde ke spuštění skriptu, který se postará o vykreslování diagramu zapojení. Po dokončení vykreslování je diagram automaticky obnoven. Doba vykreslování závisí na počtu zapojených síťových prvků v lokální síti. Čím více je síťových prvků zapojeno, tím delší bude doba vykreslování. Důvodem je průběh dotazování skriptu každého připojeného síťového prvku na jeho sousedy. Diagram zapojení obsahuje po

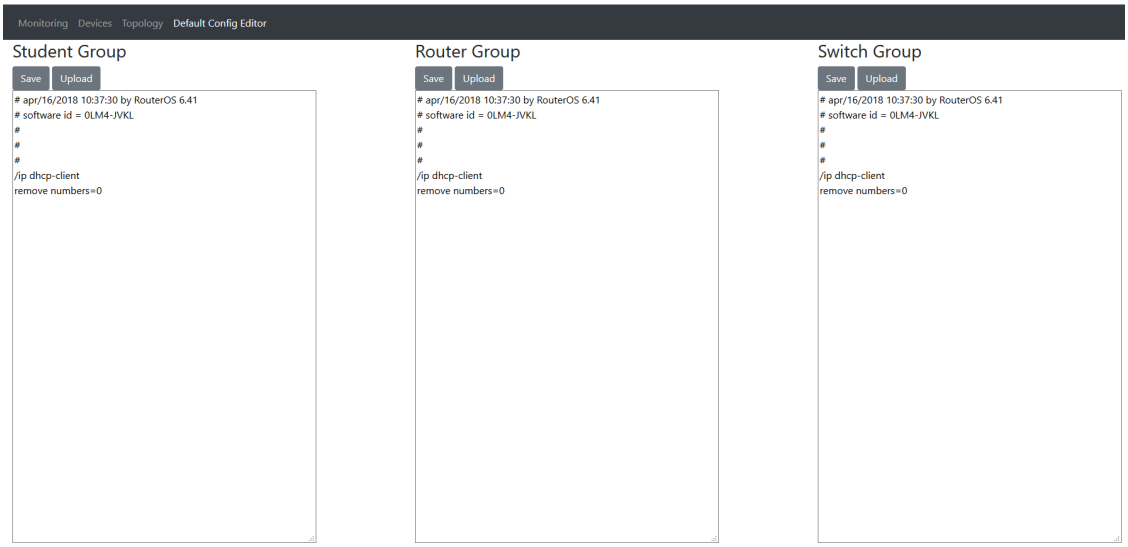
vykreslení známé i neznámé síťové prvky s nakonfigurovanou IP adresou. Rozhraní je zobrazeno na obr. 4.5.



Obr. 4.5: Rozhraní Topology.

4.3.4 Default Config Editor

Rozhraní s názvem Default Config Editor obsahuje tři textové editory pro skupiny Student, Router a Switch. Rozhraní je zobrazeno na obr. 4.6.



Obr. 4.6: Rozhraní Default Config Editor.

V každém textovém editoru je možné libovolně upravovat konfigurační soubory síťových prvků. Tlačítko Save umožňuje provedené změny uložit na webový server.

Tlačítkem Upload dojde k nahrání konfiguračního souboru na všechny síťové prvky, které patři pod danou skupinu. Provedené změny se na síťových prvcích ihned projeví.

4.4 Modifikace webové aplikace

Zdrojové kódy webové aplikace jsou volně přístupné a mohou být libovolně upravovány dle potřeb uživatele. Složka projektu webové aplikace obsahuje celkem čtyři Django aplikace, které jsou pojmenovány jako Monitoring, Discovery, Topology a Editor. Každá Django aplikace zajišťuje funkcionalitu jednotlivých rozhraní webové aplikace. Uvnitř Django aplikací se poté nacházejí zdrojové kódy vytvořených skriptů, moduly frameworku Django a složka templates, ve které jsou uloženy šablony webové aplikace. Přidání nových funkcí do rozhraní webové aplikace lze obecně provést prostřednictvím modifikace příslušné šablony a modulů views.py a urls.py.

4.4.1 Přidávání nových funkcí

Novou funkci je nejprve potřeba implementovat a namapovat ji na URL webové aplikace. K tomuto účelu slouží moduly views.py a urls.py. Následně je potřeba patřičně upravit šablonu příslušného rozhraní webové aplikace. Na obr. 4.7 se nachází možný způsob implementace funkce v modulu views.py

```
1  @login_required
2  @csrf_exempt
3  def funkce(request):
4      ips = []
5      if request.is_ajax():
6          set_interface = request.POST.get('set_interface')
7          jd = json.dumps(set_interface)
8          json_to_list = eval(json.loads(jd))
9          for item in json_to_list:
10             for value in item.values():
11                 ips.append(value[0])
12     for i in ips:
13         command = 'python3 /script.py --ip {}'.format(i)
14         process = subprocess.Popen(command.split())
15
16     return render(request, 'discovery/devices.html')
```

Obr. 4.7: Implementace funkce v modulu views.py.

Funkce pracuje s proměnnou datového typu list. Do této proměnné jsou prostřednictvím získaných dat z technologie Ajax ukládány IP adresy síťových prvků. Zasílání dat technologií Ajax je možné libovolně upravovat v šabloně pro rozhraní

webové aplikace. Podle počtu uložených IP adres je spouštěn uživatelem vytvořený skript, který postupně provádí konfigurace na síťových prvcích. Parametrem skriptu je v tomto případě IP adresa síťového prvku. Celá funkce pak vrací kompletní šablonu určenou pro rozhraní webové aplikace.

Dalším krokem je potřeba namapovat vytvořenou funkci na URL webové aplikace. Na obr. 4.8 se nachází způsob mapování v modulu urls.py.

```
1 urlpatterns = [
2     url(r'^funkce/$', views.funkce, name='funkce'),
3 ]
```

Obr. 4.8: Způsob mapování v modulu urls.py.

Do proměnné urlpatterns je možné mapovat vytvořené funkce v modulu views.py na příslušná URL. Nejprve je nutné specifikovat výraz, kterým se bude funkce v rámci webové aplikace spouštět. Následně je potřeba zadat cestu k funkci. Nepovinným parametrem je pak jméno namapované funkce.

Posledním krokem je implementace nově vytvořené funkce do prostředí webové aplikace. K tomuto účelu slouží šablona daného rozhraní, kde lze pomocí jazyka JavaScript a technologie Ajax volat vytvořené funkce umístěné v modulu views.py. Jak bude funkce z rozhraní webové aplikace spouštěna záleží čistě na uživateli. Na obr. 4.9 se nachází implementace funkce do rozhraní webové aplikace.

```
1  $("#button1").click(function(){
2    var formData = new FormData($('#submit_form')[0]);
3    formData.append("set_interface", JSON.stringify(devices_data));
4    $.ajax({
5      url:'/funkce/',
6      type: 'POST',
7      data: formData,
8      async: true,
9      cache: false,
10     contentType: false,
11     processData: false,
12   });
13 });
```

Obr. 4.9: Implementace funkce do rozhraní webové aplikace.

Jak lze vidět, funkce z modulu views.py je spouštěna po kliknutí na tlačítko. Záleží však čistě na uživateli, pro jakou variantu spouštění funkce se rozhodne. Základem je však použití technologie Ajax, která pomocí definovaného URL zavolá vytvořenou funkci. Součástí technologie Ajax je i schopnost na funkci zaslat data ve formátu JSON. Data jsou vždy načítána do lokální proměnné při označení zaškrtávacích polí webové aplikace. Pro funkci v modulu views.py jsou zasílána data

ve formě IP adres síťových prvků, neboť IP adresa je povinným parametrem pro spuštění uživatelem vytvořeného skriptu. Na obr. 4.10 se nakonec nachází způsob implementace tlačítka do rozhraní webové aplikace, která je realizována opět uvnitř šablony.

```
1            <input class="btn btn-secondary" type="button"
               id="button1" value="button1">
```

Obr. 4.10: Implementace tlačítka do webové aplikace.

4.4.2 Úprava automatické aktualizace

Pro automatickou aktualizaci obsahu v rozhraní webové aplikace byla rovněž využita technologie Ajax. Aktualizace obsahu probíhá na principu opakovaného volání URL funkce umístěné v modulu views.py. Na obr. 4.11 se nachází příklad implementace automatické aktualizace.

```
1 setInterval(function(){
2   $.ajax({
3     url:'/devices/',
4     method:'post',
5     data:$(this).serialize(),
6   });
7 }, 10000)
```

Obr. 4.11: Implementace automatické aktualizace.

Pomocí funkce setInterval je možné nastavit časový úsek, během kterého se bude opakovaně zasílat požadavek na URL dané funkce. Časový úsek je v tomto příkladu nastavený na hodnotu 10000, což odpovídá době T = 10 s.

4.5 Nedostatky a návrhy na vylepšení

Webová aplikace obsahuje několik nedostatků a limitací, které by mohly být námětem pro další vývoj. Prvním nedostatkem je neschopnost webové aplikace hromadně pracovat se síťovými prvky, které obsahují správcem změněné přihlašovací údaje. Webová aplikace standardně pracuje se síťovými prvky, které obsahují výchozí přihlašovací údaje známé z továrního nastavení. Tyto přihlašovací údaje jsou pro všechny síťové prvky MikroTik totožné a fungují zde jako standard pro přihlášení.

V rozhraní Devices je implementována kontrola, zda je webová aplikace schopna

se na síťový prvek přihlásit. Stav přihlášení lze sledovat ve sloupci Login v tabulce známých síťových prvků. Pokud je stav označen jako FAILED, znamená to, že správce změnil na síťovém prvku přihlašovací údaje a není tak možné síťový prvek přes webovou aplikaci spravovat. Možným řešením tohoto problému je implementace databáze, která by obsahovala přihlašovací jméno a hash hesla pro každý síťový prvek. Tyto přihlašovací údaje by bylo nutné uživatelem zadat pro každý nalezený síťový prvek zvlášť.

Druhým nedostatkem webové aplikace je typ certifikátu, který je vytvořenými skripty importován na síťové prvky. Certifikát je podepsaný webovým serverem, na kterém je webová aplikace umístěna. Toto řešení není z hlediska bezpečnosti ideální. Možným řešením je využití důvěryhodné certifikační autority, kterou by byl certifikát podepsán.

Vykreslování síťové topologie rovněž obsahuje několik limitací. Hlavní limitace je vykreslování pouze těch síťových prvků, které se nacházejí v podsíti 192.168.0.0/16. Toto omezení bylo implementováno z důvodu rychlosti vykreslování síťové topologie. Možným řešením tohoto problému je implementace vykreslování přes protokol MAC-Telnet. Tento způsob implementace by vyřešil problém s omezením. Navíc by bylo možné vykreslovat i síťové prvky, které by neměly nastavenou IP adresu na rozhraní. Tento způsob implementace však zahrnuje nestabilitu protokolu MAC-Telnet, a proto nebylo na toto řešení přistoupeno.

5 ZÁVĚR

Výsledkem této práce je návrh a vytvoření interaktivní webové aplikace pro hromadnou správu síťových prvků firmy MikroTik. V teoretické části práce byl popsán operační systém RouterOS společně s veškerými možnostmi konfigurace. Důraz byl kladen především na podrobnější popis služeb MikroTik API a její šifrované varianty API-SSL, která je v rámci řešení webové aplikace používána. Dále byl rovněž vysvětlen princip komunikace služby MikroTik API společně s popisem významu jednotlivých slov. V závěru teoretické části práce byly stručně popsány nástroje, které jsou při vývoji webové aplikace využity.

Praktická část práce popisuje návrh a průběh vývoje webové aplikace. Je zde znázorněna topologie, nad kterou byla webová aplikace vyvíjena. Během vytváření webové aplikace bylo zjištěno, že síťové prvky v továrním nastavení nemají přiřazenou IP adresu na žádném rozhraní. Rovněž nemají importovány žádné certifikáty, které jsou nutné pro správnou funkci služby API-SSL. Během vývoje bylo tedy nutné implementovat řešení, které zajistí konektivitu síťových prvků na síťové vrstvě a umožní import certifikátu.

Pro nalezení síťových prvků v lokální síti byl vytvořen skript, který za pomoci všesměrového vysílání MNDP dotazů umožňuje síťové prvky nalézt a zároveň zjistit jejich základní informace. Nalezené síťové prvky jsou postupně ukládány do vytvořeného databázového systému.

Při vytváření webové aplikace bylo rovněž nutné zajistit konektivitu pro síťové prvky bez IP adresy. Tento problém byl vyřešen pomocí MAC-Telnet protokolu, který zajišťuje připojení na síťové prvky prostřednictvím jejich MAC adresy. V souvislosti s MAC-Telnet protokolem byl dále vytvořen skript pro nastavení DHCP klienta na vybraných síťových prvcích. Následně byl popsán systém generace a importu certifikátu na síťové prvky.

Síťové prvky s nastavenou IP adresou a importovaným certifikátem již mohou být spravovány přes službu API-SSL. V rámci dalšího vývoje webové aplikace byly vytvořeny skripty, které umožňují manipulovat s konfiguračními soubory síťových prvků. V souvislosti s konfiguračními soubory byl dále vytvořen editor, přes který je možné libovolně upravovat konfigurační soubory síťových prvků. Upravené konfigurační soubory lze dále na síťové prvky nahrát, přičemž každý konfigurační soubor patří pro danou skupinu síťových prvků.

Vývoj webové aplikace rovněž zahrnoval implementaci monitorovacího systému a systému pro vykreslování síťové topologie. Bylo vysvětleno, jakým způsobem byly tyto systémy vytvářeny a jak pracují. Nakonec byl popsán vytvořený systém hromadné správy síťových prvků a byla rovněž otestována komunikace přes službu MikroTik API a API-SSL.

V závěru práce bylo vysvětleno, jakým způsobem je možné spravovat uživatelské účty webové aplikace a jak je implementováno přihlášení do systému. Rovněž byla popsána jednotlivá rozhraní webové aplikace. Nakonec byly uvedeny příklady, jakým způsobem je možné rozhraní webové aplikace modifikovat. Rovněž byly popsány nalezené nedostatky webové aplikace a návrhy pro další vývoj.

LITERATURA

[1] About us. *MikroTik Routers and Wireless - About* [online]. [cit. 2017-11-05]. Dostupné z: https://mikrotik.com/aboutus

[2] BURGESS, Dennis. *Learn RouterOS - Second Edition*. 2nd edition. Rumford: Lulu.com, 2011. ISBN 978-1-105-06959-8.

[3] Configuration. *Grafana Labs* [online]. [cit. 2018-04-11]. Dostupné z: http://docs.grafana.org/installation/configuration/

[4] ELMAN, Julia. a Mark. LAVIN. *Lightweight Django*. Sebastopol, CA: O'Reilly Media, 2015. ISBN 978-1-491-94594-0.

[5] Grafana Documentation. *Grafana Labs* [online]. [cit. 2018-04-11]. Dostupné z: http://docs.grafana.org/

[6] InfluxDB 1.5 documentation. *InfluxData* [online]. [cit. 2018-04-11]. Dostupné z: https://docs.influxdata.com/influxdb/v1.5/

[7] JAY A. KREIBICH. *Using SQLite*. Sebastopol, CA: O'Reilly, 2010. ISBN 978-059-6521-189.

[8] LUTZ, Mark. *Learning Python*. Fifth edition. Beijing: O'Reilly, 2013. ISBN 978-1-449-35573-9.

[9] MAC access. *MAC access - MikroTik Wiki* [online]. [cit. 2017-11-05]. Dostupné z: https://wiki.mikrotik.com/wiki/MAC_access

[10] Manual:API. *Manual:API - MikroTik Wiki* [online]. [cit. 2017-11-05]. Dostupné z: https://wiki.mikrotik.com/wiki/Manual:API

[11] Manual:API-SSL. *Manual:API-SSL - MikroTik Wiki* [online]. [cit. 2017-11-05]. Dostupné z: https://wiki.mikrotik.com/wiki/Manual:API-SSL

[12] Manual:Hotspot HTTPS example. *Manual:Hotspot HTTPS example - MikroTik Wiki* [online]. [cit. 2017-11-05]. Dostupné z: https://wiki.mikrotik.com/wiki/Manual:Hotspot_HTTPS_example

[13] Manual:License. *Manual:License - MikroTik Wiki* [online]. [cit. 2017-11-05]. Dostupné z: https://wiki.mikrotik.com/wiki/Manual:License

[14] Manual:Webfig. *Manual:Webfig - MikroTik Wiki* [online]. [cit. 2017-11-05]. Dostupné z: https://wiki.mikrotik.com/wiki/Manual:Webfig

[15] Manual:Winbox. *Manual:Winbox - MikroTik Wiki* [online]. [cit. 2017-11-05]. Dostupné z: https://wiki.mikrotik.com/wiki/Manual:Winbox

[16] NIXON, Robin. *Learning PHP, MySQL and JavaScript*. Fourth edition. Sebastopol, CA: O'Reilly Media, 2014. ISBN 978-1-491-91866-1.

[17] Telegraf 1.5 documentation. *InfluxData* [online]. [cit. 2018-04-11]. Dostupné z: https://docs.influxdata.com/telegraf/v1.5/

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CSS	Cascading Style Sheets
DHCP	Dynamic Host Configuration Protocol
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IP	Internet Protocol
JSON	JavaScript Object Notation
OID	Object Identifier
PBKDF2	Password-Based Key Derivation Function 2
MAC	Media Access Control
MNDP	MikroTik Neighbor Discovery Protocol
SFTP	Secure File Transfer Protocol
SHA-256	Secure Hash Algorithm - 256
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UID	Unique Identifier
URL	Uniform Resource Locator

SEZNAM PŘÍLOH

A PŘÍLOHA

A.1 Obsah přiloženého CD

Na přiloženém CD je, kromě elektronické verze ve formátu pdf, umístěn kompletní projekt webové aplikace. Uvnitř projektu se nacházejí veškeré zdrojové kódy. Celý projekt je umístěn v archivu priloha.zip, který je opatřený heslem. Webovou aplikaci je možné vyzkoušet na webovém serveru, na který se lze z vnější sítě připojit prostřednictvím VPN. IP adresy, heslo k archivu a veškeré přihlašovací údaje lze získat na emailové adrese xkloda00@stud.feec.vutbr.cz.